4. Sonderausgabe

# DOS EXTRA

18,– DM 145 Ös 18,– sfr

Sonderausgabe DOS-EXTRA

Nr. 4 1988

ISSN 0934 – 2842

DMV-Verlag

Eine Publikation der
DOS INTERNATIONAL

## Speicherresidente Programme

- Grundlagen
- Utilities
- Anwendung

## Projekt

Desktop Publishing zum Abtippen

## Grafik

- Grundlagen
- EGA-Karte selbst programmiert

## MS-DOS Intern

- Festplattendoktor
- Tastatureingaben automatisiert
- MS-DOS-Interrupt 21

Jede Menge Tips & Tricks zu MS-DOS

# Silbentrennung perfekt

Mal ehrlich: Trauen Sie eigentlich der Silbentrennung Ihres Textprogramms? Oder trennen Sie doch lieber von Hand? Sicherheitshalber, versteht sich.

Nun, dann haben Sie ab sofort eine Sorge weniger. Steigen Sie um auf TextMaker! Die brandneue **Version 1.2** besitzt eine ganz außergewöhnlich treffsichere deutsche Silbentrennung. Wo andere Textprogramme sich alle eineinhalb Seiten irren, macht TextMaker oft nur alle 15 Seiten einen Fehler!

Sicher, auf den ersten Blick vielleicht eine Nebensächlichkeit. Aber sie zeigt, daß wir bei der Entwicklung von TextMaker stets an unsere Kunden denken. Denn nichts kommt schlechter an, als ein Brief mit falschen Worttrennungen.

TextMaker ist eine ausgesprochen flexible Textverarbeitung. Sie steuern TextMaker über Pulldown-Menüs oder mit Hilfe des WordStar-kompatiblen Befehlssatzes. Ganz wie Sie wollen. Oder Sie definieren sich Ihren ganz persönlichen Befehlssatz. Denn Sie können die Tasten ALT-A bis ALT-Z sowie alle Funktionstasten auf vier Ebenen mit Befehlen oder Texten belegen. Dadurch verhält sich TextMaker immer so, wie es Ihnen gefällt.

Einige der Vorzüge von TextMaker 1.2:

- Texte in bis zu neun Textfenstern gleichzeitig bearbeiten
- Textgröße nur durch den RAM-Speicher begrenzt
- Der integrierte Druckerspooler druckt, während Sie schreiben
- 63 Tastaturmakros für Texte und Befehle
- Rechnen im Text mit 26 Variablen
- Adressenverwaltung und Serienbrieffunktion
- Phonetisches Suchen: Mayer? Meier? Maier?
- Die UNDO-Taste rettet bis zu 32000 Zeilen
- Layout auf dem Bildschirm zeigen
- Drucken auf Bildschirm: *kursiv*, **fett**, unterstrichen ...
- Integrierter Dateimanager: Löschen, Kopieren, Umbenennen, Formatieren
- Zwischendurch DOS-Befehle eingeben und andere Programme starten
- Einblendbare ASCII-Tabelle mit dem kompletten IBM-Zeichensatz
- Stichwortverzeichnisse, Inhaltsverzeichnisse und Fußnoten
- Automatische Textsicherung in beliebigen Zeitabständen
- Unterstützung für alle Drucker, mehr als 60 Druckertreiber
- Für IBM-PCs und Kompatible mit mindestens 256 KByte RAM
- Für CGA-Farbgrafik, Monochrom-Karte, MCGA und Hercules
- Jetzt auch komplette Unterstützung von EGA- und VGA-Karten
- Auf Wunsch auch im 3,5″-Diskettenformat lieferbar

Fordern Sie unsere Demo-Diskette an! Für einen 10-Mark-Schein können Sie TextMaker gründlich testen und sich begeistern lassen. Die Demo-Version enthält den vollständigen TextMaker bis auf die Funktionen »Drucken« und »Speichern«. Und Sie können die Silbentrennung von TextMaker ausprobieren. Sicherheitshalber, versteht sich.

*Bitte Coupon ausfüllen, ausschneiden und an SoftMaker schicken.*

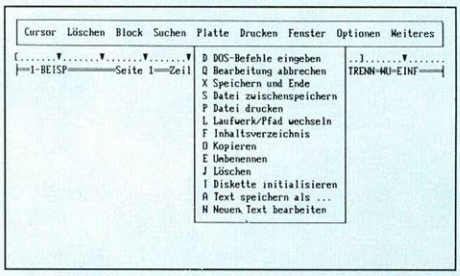
☐ Bitte schicken Sie mir erst einmal kostenlos Informationen.

☐ Senden Sie mir eine Demodiskette. Einen 10 Mark-Schein lege ich bei.

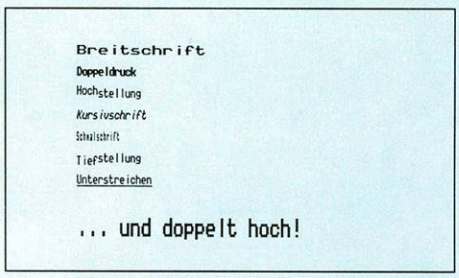
☐ TextMaker ist genau, was ich suche! Schicken Sie mir ein Exemplar!

Anschrift:

Unterschrift:




*Übersichtliche Pulldown-Menüs*




*Druckbild bereits auf dem Monitor*

## nur DM 148.-

Unverbindliche Preisempfehlung nur DM 148,– bei Bezug über Ihren Händler. Bei Versand DM 158,– inklusive Porto und Verpackung.
Lieferung auf Verrechnungsscheck oder Nachnahme.

TextMaker erhalten Sie zum Beispiel bei:

MEGA /// Team
Kirchhellener Straße 262
4250 Bottrop
☎ 02041/94842

U. Trommer
Rathausstraße 32
5248 Wissen
☎ 02742/1465

Bothe EDV
Schleißheimer Straße 201
8000 München 40
☎ 089/3084152

# SoftMaker

Grabbestraße 9
8500 Nürnberg 90
Telefon 0911/303333

Händleranfragen stets willkommen!

# Informationen ohne Ende

»Wo bekommt Ihr nur Euer ganzes Material für die Artikel her?« werden wir immer wieder von unseren Lesern gefragt. Da gibt es monatlich die mit Informationen prall gefüllte DOS International, neuerdings ebenfalls monatlich die Window mit einem breiten Angebot für Software-Anwender, und dann auch noch alle drei Monate ein dickes DOS Extra – und trotzdem gehen die Themen nicht aus. Der Grund dafür liegt nicht etwa darin, daß der Personalcomputer, das Betriebssystem MS-DOS oder die PC-Software besonders erklärungsbedürftig sind, sondern an der Flexibilität der PC-Hardware (Steckkartenkonzept), der einfachen Ausbaufähigkeit von MS-DOS (Benutzeroberflächen, TSR-Programme etc.) und der Vielfalt des Softwareangebotes (das größte der Welt), so daß sich immer wieder neue, interessante Aspekte finden, über die zu berichten sich lohnt.

An der Informationsvielfalt haben auch Sie als Leser mit Ihren Einsendungen in Form von Programmen, Utilities sowie Tips und Tricks einen entscheidenen Anteil. Deshalb an dieser Stelle gleich ein herzliches Dankeschön und die Bitte, mit der Programmier-Freudigkeit nicht nachzulassen. Wir tun jedenfalls unser Bestes, um Sie mit neuen Informationen zu versorgen, die Ihnen bei Ihren neuen Projekten weiterhelfen. So bietet Ihnen auch die vierte DOS Extra eine bunte Mischung von Informationen rund um den PC.

Wenn Sie sich schon oft über das umständliche Editieren in der MS-DOS-Kommandoebene geärgert haben, können Sie jetzt erleichtert aufatmen, denn diese Ausgabe der DOS Extra präsentiert Ihnen einen Kommandoeditor, der den Editierkomfort erhöht, sich die letzten acht eingegebenen Befehle merkt und noch einiges mehr kann.

Die Programmierer kommen bei dem Schwerpunkt, der sich mit speicherresidenten Programmen befaßt, voll auf ihre Kosten, denn es wird nicht nur erklärt, wie diese Programme theoretisch funktionieren, sondern Sie finden auch konkrete Beispiele zum Nachvollziehen.

Für die an dem Thema Desktop Publishing interessierten Leser enthält dieses Sonderheft einen besonderen Leckerbissen: »SDTP«, ein komplettes Desktop-Publishing-Programm zum Abtippen. Wenn es sich auch nicht mit Spitzenprogrammen, wie dem Pagemaker oder Ventura Publisher messen kann, so ist es doch für kleinere Aufgaben wie Glückwunschkarten, Einladungen oder Handzettel hervorragend geeignet. Weil das Programm auch im Quellcode vorliegt, können Sie es problemlos erweitern und an Ihre persönlichen Bedürfnisse anpassen.

An die Freunde der Grafik wurde in dieser DOS-Extra-Ausgabe ebenfalls gedacht. Ein ausführlicher Grundlagenbeitrag beschäftigt sich mit dem Monochromadapter, der Hercules-Grafikkarte, der CGA-Farbgrafikkarte und der EGA-Karte. Wer eine EGA-Karte besitzt, für den haben wir ein Programm, mit dem sich verschiedene Cursor- und Schriftarten programmieren lassen.

Die Freaks unter den Programmierern werden ausführlich über den MS-DOS-Interrupt 21hex informiert, so daß sie fortan noch schnellere Programme schreiben können.

Natürlich, und das ist bei der DOS Tradition, bietet Ihnen auch dieses Heft wieder eine Menge Tips und Tricks. So erfahren Sie zum Beispiel, wie Sie Tastatureingaben programmieren ohne die Tastatur zu benutzen. Und das ist nur ein Beispiel aus dem großen Angebot.

Schließlich haben die vielen Anrufe und Zuschriften uns nicht ruhen lassen, das Clock-Utility aus der ersten DOS Extra zu überarbeiten und auch an den Hercules-Standard anzupassen, so daß es jetzt auf allen Standard-Grafikkarten läuft.

Nun wünschen wir Ihnen wieder viel Spaß und neue Erkenntnisse bei der Lektüre der vierten DOS Extra.

Ihr

Martin Aschoff

## Tips zur Eingabe:

Checksummer 6
– *Geprüfte Eingabe*

## MS-DOS intern:

Generaluntersuchung mit dem Festplattendoktor 10

Optimierung mit System 14
– *»config.sys«*

Programmieren wie von Zauberhand 16
– *Tastaturpuffer programmiert*

Tastatureingaben gespart 18
– *Assembler-Utility*

Der MS-DOS-Interrupt 21hex (Teil 1) 19
– *Grundlagen*

Der MS-DOS-Interrupt 21hex (Teil 2) 30
– *Grundlagen*

## Speicherresidente Programme:

Speicherresidente Programme – Zauberlehrlinge im Hintergrund 40
– *Grundlagen*

Clock-Utility für alle Grafikkarten 50
– *speicherresidente Uhr*

Befreien Sie Ihren Speicher 60
– *speicherresidente Programme löschen*

Einblick in den Speicher – residente Programme aufgespürt 70

## Desktop Publishing:

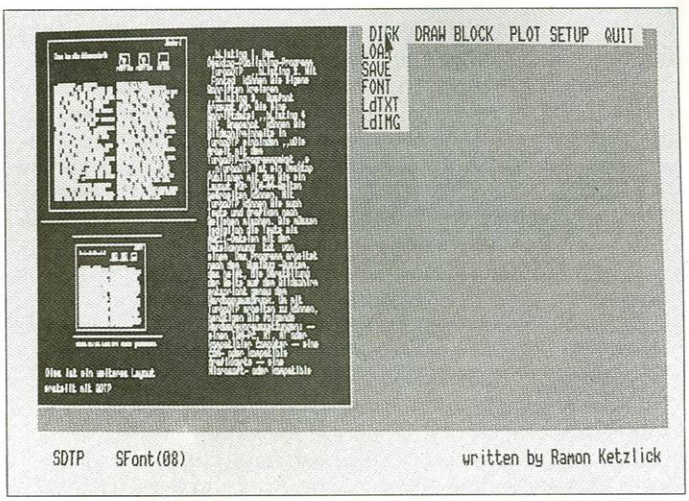
Ein Weg zum gekonnten Layout 76
– *Grundlagen*

Desktop Publishing zum Abtippen 78
– *Listing*

Layouts bearbeiten mit SDTP 104
– *So wird's gemacht*

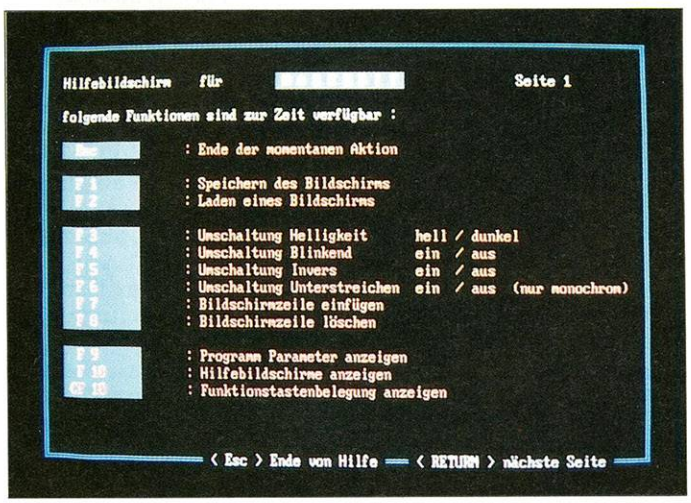
**Eine Uhr, die ständig auf dem Bildschirm die Zeit anzeigt ist nicht nur für Programmierfreaks eine nützliche Einrichtung. Die Uhrzeit will schließlich jeder korrekt wissen.**

**Seite 50**



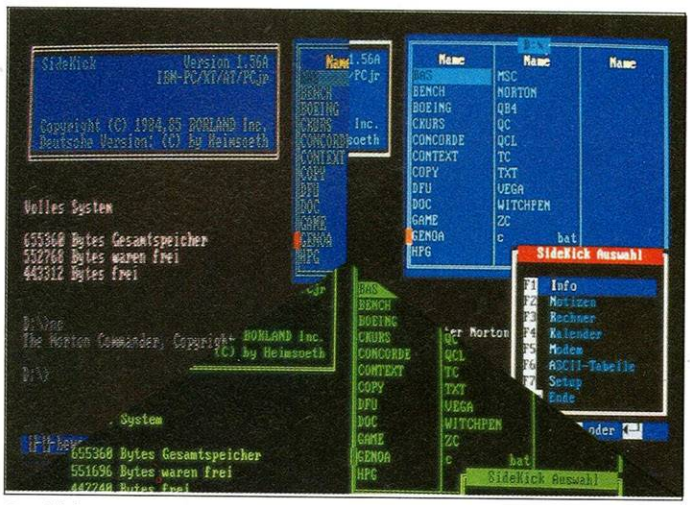

**SDTP heißt ein waschechtes Desktop-Publishing-Programm zum Abtippen. Preiswerter geht es wirklich nicht mehr.**

**Seite 104**




**Hilfsbildschirme erleichtern die Arbeit mit unbekannter Software. Rüsten auch Sie Ihre Programme damit aus.**

**Seite 140**




**Grafikkarten sind nicht nur für Computer-Bilder zuständig. Auch die Textdarstellung hängt von der richtigen Hardware ab.**

**Seite 164**

```
A:>DEBUG COMMAND1.COM
-R
AX=0000 BX=0000 CX=5DEC DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=43E6 ES=43E6 SS=43E6 CS=43E6 IP=0100 NV UP EI PL NZ NA PO NC
43E6:0100 E97D0C    JMP  0D80
-S100 5EEC 'AUTOEXEC.BAT'
43E6:14BE
-D14B0 L20
43E6:14B0 4D 4D 41 4E 44 2E 43 4F-4D 00 00 00 3A 5C 41 55
MMAND.COM...:\AU
43E6:14C0 54 4F 45 58 45 43 2E 42-41 54 00 FF 00 00 00 00
TOEXEC.BAT......
-S100 5EEC BA BB 14
43E6:122C
-U122C L8
43E6:122C BABB14    MOV  DX,14BB
43E6:122F B8003D    MOV  AX,3D00
43E6:1232 CD21 INT  21
-A122C
43E6:122C CALL 5EEC
43E6:122F
-W
```

**Mit dem Debugger und einem Assembler-Listing können Sie Ihre Command-Datei so patchen, daß sie eine Paßwortabfrage enthält.**

**Seite 124**

**Mit dem Interrupt 21hex lassen sich fast alle Funktionen des Betriebssystems aufrufen. Wie das funktioniert, lesen Sie auf**

**Seite 19**

## Tips & Tricks:


Tips & Tricks zu Pascal **108**
- Editor statt Read-Befehl in Turbo Pascal

Tips & Tricks zu Basic **110**
- *Feststellen des Computertyps*
- *Abfrage der Sondertasten*
- *Feststellen, ob der Diskettenmotor läuft*
- *Die momentane Position des Schreib-/Lesekopfes*
- *Diskette auf Schreibschutz überprüfen*
- *Zahl der angeschlossenen Laufwerke*
- *Größe des RAM*
- *Softwaregesteuertes Setzen der Tastatur-Modi*
- *Sperren der Tastatur*
- *Tastaturpuffer löschen*
- *Umgehen des Listschutzes beim Basic-Interpreter*
- *Unterdrücken von »Break« und »Ctrl-C«*
- *Softwaregesteuerte Hardcopy*

Tips & Tricks für C-Programmierer **112**
- *Gesetzte Bits feststellen*
- *Turbo C und die Compiler-Optimierungseinstellungen*
- *Verwendung von »printf«, »write(str)« und »puts(putch)«*
- *Cursorgesteuerte Programmauswahl*
- *Systeminformationen mit Turbo C*


## Utilities:


PC-Paßwortschutz **124**

Kommando-Editor für MS-DOS **130**

Hilfsbildschirme – ein Service, der viel bringt **140**

Fenstertechnik mit Turbo C **118**


## Grafik:


Verschiedene Cursor wie bei GEM **152**

Alles über Grafikkarten **164**
- *Grundlagen*


## Rubriken:


Editorial **3**

Impressum **170**

Inserenten **170**

# Check-summer

**Mit SCheck, dem Checksummenprüfprogramm der DOS International, werden alle Listings auf korrekte Eingabe überprüft. Der eingebaute Editor erlaubt sofortige Fehlerkorrektur. Sie sparen damit jede Menge Zeit beim Abtippen von Listings.**

Immer mehr Leser der DOS International kaufen sich keine fertige Software, sondern finden in der Zeitschrift Programme abgedruckt, die Ihren Wünschen voll entsprechen.

Damit nun aber die Eingabe möglichst schnell und fehlerfrei erfolgen kann, finden Sie in der DOS sämtliche Listings mit Prüfsumme versehen. Mit Hilfe der Routine SCheck (Listing 1) können Sie Ihre Zeilen mit dem abgedruckten Programm auf Fehler überprüfen. SCheck stellt Ihnen dazu zwei verschiedene Eingabevarianten zur Verfügung.

Beim ersten Weg geben Sie mit dem Editor der entsprechenden Programmiersprache – oder auch mit einem beliebigen Texteditor, der ASCII-Dateien erzeugt – nur den Quellcode ein. Dieser steht hinter der Prüfsumme (in eckigen Klammern) und der Zeilennummer (Zahl mit Doppelpunkt). Listing 2 zeigt eine Pascal-Routine, wie sie im Heft abgedruckt ist, Listing 3 die gleichen Programmzeilen, so wie Sie sie in den Computer eingeben müssen. Eine Programmzeile wird immer durch die Zeilennummer angezeigt und kann dabei im Druck über mehrere Zeilen gehen. In solch einem Fall müssen Sie die Anweisungen durchgängig in einer Zeile in den Editor eintippen.

Als nächstes starten Sie SCheck mit dem Dateinamen

```
scheck <Name.Ext>
```

Legen Sie sich bitte von der eingegebenen Datei eine Kopie an. Zwar arbeitet SCheck nicht mit dem Original, sondern mit einer Kopie, die erst ganz zum Schluß als aktuelle Datei verändert wird, doch wenn ein Fehler beim Speichern auftritt, so ist es nicht ausgeschlossen, daß die mühsam eingetippten Daten verloren gehen. Also immer als erstes eine Kopie der Daten anlegen.

Das Bild zeigt die aktuelle Zeile in dem oberen Fenster. Die Checksumme wird darunter links angezeigt. Diese vergleichen Sie mit der abgedruckten Zahl in den eckigen Klammern. Rechts daneben steht die Zeilennummer und der Name der aktuellen Datei. Im unteren Fenster werden Fehlermeldungen und Hinweise auf die weitere Bearbeitung ausgegeben. Im hier abgebildeten Fall können Sie mit [Return] die Zeile unverändert übernehmen oder mit [E] den Editor aufrufen.

Der Editor dient der Korrektur fehlerhafter Zeilen. Er arbeitet immer im Einfügemodus, so daß fehlerhafte Eingaben extra gelöscht werden müssen. Sie können also nichts versehentlich überschreiben. Mit den Cursortasten steuern Sie den Cursor im Editierfenster. [Home] bringt den Cursor an den Anfang der aktuellen Programmzeile und [End] ans Ende. Löschen können Sie das Zeichen links vom Cursor mit der Taste oberhalb der Enter-Taste (auf der Schreibtastatur). [Del] löscht das Zeichen genau an der Cursorposition. Mit [Return] übernehmen Sie die geänderte Programmzeile. Die Leerzeichen vor der ersten gültigen Anweisung einer Programmzeile werden übrigens bei der Prüfsummenberechnung nicht berücksichtigt. Strukturen können Sie somit nach eigener Wahl einbauen. Weiter hinten stehende Leerzeichen sind für die Prüfsummenberechnung allerdings relevant. Die Tabulatorfunktion dürfen Sie nicht verwenden, wenn Sie mit der Prüfsumme korrekt arbeiten wollen.

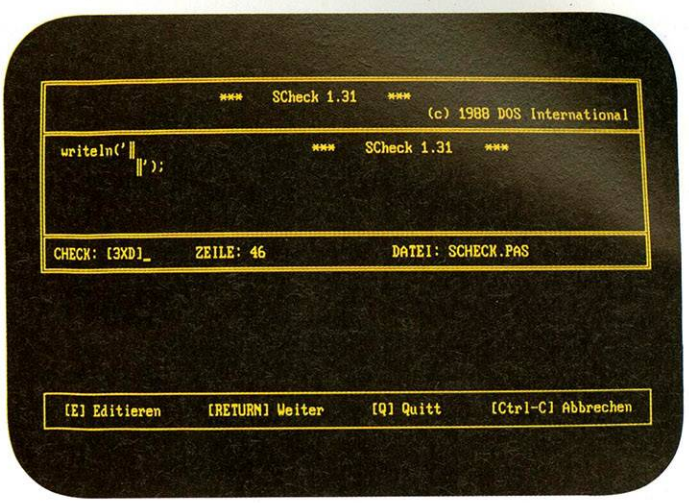



*Bild. Mit SCheck können Sie auch falsche Zeilen korrigieren*

Haben Sie so das gesamte Programm durchgearbeitet, so speichert ein letzter Druck auf [Enter] das korrigierte Listing. Das Programm wird beendet und das fehlerfreie Programm steht für die weitere Bearbeitung mit einem Compiler, Assembler oder Interpreter zur Verfügung.

Listing 4 zeigt ein Basic-Programm, wie es in der DOS International abgedruckt wird. Die Eingabe erfolgt wiederum ohne Prüfsumme, aber in diesem Fall mit Zeilennummer. Listing 5 zeigt, wie die Routine eingegeben werden muß.

SCheck erlaubt aber noch einen zweiten Weg, das Listing korrekt einzugeben. Diesmal geben Sie die Prüfsumme mit ein. Starten Sie den Programmeditor (wieder gilt, daß die Dateien nur aus ASCII-Werten bestehen dürfen) und geben Sie das Programm mit der Prüfsumme ein. Die eckigen Klammern lassen Sie weg und auch die Zeilennummer. Listing 6 zeigt, wie die Datei mit dem Programm aus Listing 2 aussehen muß. Listing 7 demonstriert das Ganze für das Basic-Programm aus Listing 4.

SCheck starten Sie jetzt mit

```
scheck <Name.Ext> /a
```

Den Schrägstrich »/« können Sie auch durch » – « ersetzen. Desgleichen darf die Option auch als Großbuchstabe angegeben werden. Der Checksummer verwandelt das Programm nun in eine Quelldatei für den entsprechenden Compiler. Dazu werden die ersten vier Zeichen abgeschnitten, die Prüfsumme berechnet und bei einem Fehler die entsprechende Zeile ausgegeben. Der Editor steht dann wieder zur Verfügung. An der Art, wie die Datei verändert wird, sehen Sie, daß hinter der Prüfsumme (ohne Klammern eingegeben) immer ein Leerzeichen stehen muß. Danach folgt der Programmtext (nur in Basic mit Zeilennummer).

Nach dem Durchlauf von SCheck steht wieder das fehlerfreie Programm ohne Prüfsumme auf der Diskette. Und wieder gilt, daß Sie nicht mit dem Original Ihrer stundenlangen Eintipparbeit arbeiten sollten, sondern nur mit einer Kopie derselben.

SCheck verfügt aber noch über eine weitere Option. Die Hexdump-Listings können Sie jetzt nämlich auch ohne den Debugger eingeben. Sie tippen ein Hexdum-Listing wieder mit dem Editor ein und starten dann SCheck ohne Option. Nach

```
scheck <Name.Ext>
```

wird der Text wie im ersten Fall zeilenweise mit Prüfsumme ausgegeben. Der Editor steht wiederum zur Korrektur zur Verfügung.

Befindet sich das Programm korrekt auf der Diskette, so wird SCheck mit

```
scheck <Name.Ext> /b
```

erneut gestartet. Für die Option gilt das schon oben Gesagte. Auf der Diskette wird nun eine Binärdatei mit der Erweiterung ».bin« erzeugt. Für direkt ausführbare Com-Dateien müssen Sie das Programm nur noch mit

```
ren <Name>.bin <Name>.com
```

umbenennen.
Doch auch bei dieser Eingabe werden Sie durch Prüfsummen direkt unterstützt. Geben Sie den Programmtext mit Prüfsumme ein und starten Sie den Checksummer mit

```
scheck <Name.Ext> /a
```

so werden die eingegebenen Daten zeilenweise überprüft. Die so erzeugte Datei entspricht dem Hexdump ohne Prüfsumme.

```
scheck <Name.Ext> /b
```

erzeugt die entsprechende Binärdatei.
Mit SCheck wird das Eingeben der Programme aus der DOS International bedeutend komfortabler und weniger fehlerträchtig. Wer allerdings kein Turbo Pascal besitzt, um das Programm zu starten, der findet auf der Databox zu der DOS Extra 2 und auf den Databoxen der DOS International ab der Ausgabe 3'88 das Programm als lauffähige Com-Datei. Ihren Listingeingaben steht nichts mehr im Wege.

(ev/hg)

```
[BE5]    0: program scheck;
[000]    1:
[BTD]    2: { Ultra-Check-Utility von DOS International
            }
[000]    3:
[0N7]    4: { Volker Everts, 10.1.88    }
[DF5]    5: { Version 1.4               }
[7Z7]    6: { Letzte Änderung: 13.4.88 }
[000]    7:
[000]    8:
[VM1]    9: const
[Y3E]   10:   ZEICHEN : string[32] = '0123456789ABCDEFG
           HKLMNPRSTUVWXYZ';
[H42]   11:   SPACE = #32;
[RG1]   12:   BS    = #8;
[DL1]   13:   TAB   = #9;
[XL1]   14:   BEL   = #7;
[WP1]   15:   CR    = #13;
[D32]   16:   EMPTY = '';
[RU1]   17:   ESC   = #27;
[000]   18:
[A21]   19: type
[B67]   20:   ZeilenTyp = string[255];
[000]   21:
[EM0]   22: var
[VZ4]   23:   Zeile: ZeilenTyp;
[D56]   24:   ps, check: string[5];
[26A]   25:   Name, TempName, Option: string[40];
[CN4]   26:   Ext: string[4];
[GK4]   27:   t, temp: text;
[595]   28:   f: file of byte;
[5V7]   29:   Nummer, b, code: integer;
[9N2]   30:   by: byte;
[2D2]   31:   ch: char;
[3T7]   32:   weiter, Auto, bin: boolean;
[000]   33:
[000]   34:
[BD5]   35: procedure init;
[000]   36:
[EM0]   37: var
[W33]   38:   i: integer;
[000]   39:
[BH1]   40: begin
[XV3]   41:   Auto := false;
[VX2]   42:   bin := false;
[R04]   43:   window(1,1,80,25);
[YE5]   44:   ClrScr; writeln;
[X4V]   45:   writeln('┌──────────────────────────────
              ──────────────────────────────┐');
[X7E]   46:   writeln('│                                  ***    S
           -Check 1.4    ***
              │');
[Y2H]   47:   writeln('│
                                (c) 1988 DOS Internationa
           l │');
[35V]   48:   writeln('├──────────────────────────────
              ──────────────────────────────┤');
[EAC]   49:   writeln('│
              │');
[EAC]   50:   writeln('│
              │');
[EAC]   51:   writeln('│
              │');
[EAC]   52:   writeln('│
              │');
[EAC]   53:   writeln('│
              │');
[35V]   54:   writeln('├──────────────────────────────
              ──────────────────────────────┤');
[VCE]   55:   writeln('│ CHECK:              ZEILE:
                      DATEI:
              │');
```

```
[V4V]   56:   writeln('└──────────────────────────────
              ──────────────────────────────┘');
[E03]   57:   GotoXY(1,20);
[2SS]   58:   writeln('┌──────────────────────────────
              ──────────────────────────────┐');
[0AC]   59:   writeln('│
              │');
[EPS]   60:   writeln('└──────────────────────────────
              ──────────────────────────────┘');
[587]   61:   if ParamCount <> 0 then
[H77]   62:     for i := 1 to ParamCount do
[BH1]   63:       begin
[XE6]   64:         Option := ParamStr(i);
[9V3]   65:         if (Option = '-a')
[7W3]   66:         or (Option = '-A')
[NW3]   67:         or (Option = '/a')
[2B9]   68:         or (Option = '/A') then Auto := tru
           e
[A11]   69:         else
[9V3]   70:         if (Option = '-b')
[7W3]   71:         or (Option = '-B')
[NW3]   72:         or (Option = '/b')
[6H8]   73:         or (Option = '/B') then bin := true
[L45]   74:         else Name := Option;
[DL0]   75:       end
[WC4]   76:     else Name := EMPTY;
[X52]   77:   repeat
[5N5]   78:     if Name = EMPTY then
[BH1]   79:       begin
[H03]   80:         GotoXY(3,21);
[YKB]   81:         write('Geben Sie einen Dateinamen e
           in');
[T93]   82:         GotoXY(54,12);
[503]   83:         BufLen := 24;
[UM2]   84:         read(Name);
[H03]   85:         GotoXY(3,21);
[9P5]   86:         write('
              ');
[TU0]   87:       end;
[BL4]   88:     assign(t, Name);
[CB5]   89:     {$I-} reset(t); {$I+}
[D25]   90:     if IOresult > 0 then
[BH1]   91:       begin
[1Y3]   92:         for i := 1 to 3 do
[BH1]   93:           begin
[M93]   94:             GotoXY(10,21);
[E49]   95:             write('+++++  D A T E I F E H L
           E R  +++++', BEL);
[ME2]   96:             delay(200);
[M93]   97:             GotoXY(10,21);
[086]   98:             write('
                      ');
[ME2]   99:             delay(200);
[TU0]  100:           end;
[GE3]  101:         Name := EMPTY;
[T93]  102:         GotoXY(54,12);
[ZV4]  103:         write('                          ');
[TU0]  104:       end;
[YM4]  105:   until Name <> EMPTY;
[YNF]  106:   for i := 1 to length(Name) do Name[i] :=
           UpCase(Name[i]);
[F74]  107:   i := pos('.',Name);
[113]  108:   if i = 0 then
[BH1]  109:     begin
[CT3]  110:       Name := Name + '.';
[3P4]  111:       i := length(Name);
[TU0]  112:     end;
[GF5]  113:   Ext := copy(Name, i, 4);
[U66]  114:   Name := copy(Name, 1, i-1);
[T93]  115:   GotoXY(54,12);
[WT4]  116:   write(Name, Ext);
[726]  117:   TempName := Name + '.$$$';
[WR6]  118:   assign(temp, TempName);
[5L4]  119:   rewrite(temp);
[US2]  120:   if bin then
[BH1]  121:     begin
```

```
[T56]  122:       assign(f, Name + '.BIN');
[UA3]  123:       rewrite(f);
[XW3]  124:       auto := false;
[TU0]  125:     end;
[HX2]  126:   Nummer := 0;
[TU0]  127: end;
[000]  128:
[000]  129:
[000]  130:
[BZA]  131: procedure Pruefe(var z: ZeilenTyp);
[000]  132:
[U6M]  133: { Prüfsumme berechnen. Führende Leerzeichen
            und Tabulatoren }
[K4L]  134: { werden nicht berücksichtigt. Bei Steuerze
          ichen im Text    }
[DXB]  135: { erfolgt eine Warnung
                           }
[000]  136:
[SC6]  137: var a, p, i, j: integer;
[000]  138:
[BH1]  139: begin
[FD1]  140:   p := 0;
[6D1]  141:   a := 1;
[9RB]  142:   while (z[a] = SPACE) or (z[a] = TAB) do a
          := a + 1;
[BN6]  143:   for i := a to length(z) do
[XG4]  144:     if z[i] < #32 then
[6Y3]  145:       for j := 1 to 3 do
[BH1]  146:         begin
[H03]  147:           GotoXY(3,21);
[43F]  148:           write(' +++++ WARNUNG: Steuerzeic
          hen in Zeile !          ', BEL);
[ME2]  149:           delay(200);
[H03]  150:           GotoXY(3,21);
[AS6]  151:           write('
                        ');
[ME2]  152:           delay(200);
[DL0]  153:         end
[ZY9]  154:     else p := ((i-a+1) and 7) * ord(z[i]) +
          p;
[582]  155:   ps := '     ';
[MM2]  156:   ps[1] := '[';
[AY3]  157:   for i := 2 to 4 do
[BH1]  158:     begin
[WP7]  159:       ps[i] := ZEICHEN[(p and 31) + 1 ];
[TG2]  160:       p := p shr 5;
[TU0]  161:     end;
[AN2]  162:   ps[5] := ']';
[TU0]  163: end;
[000]  164:
[000]  165:
[46A]  166: procedure edit(var z: ZeilenTyp);
[000]  167:
[EM3]  168: const s = #32;
[000]  169:
[EM0]  170: var
[B43]  171:   x: integer;
[2D2]  172:   ch: char;
[000]  173:
[598]  174: procedure schreibe(c: char);
[BH1]  175: begin
[D29]  176:   if length(z) = 255 then write(BEL)
[A11]  177:   else
[BH1]  178:     begin
[NA4]  179:       insert(c, z, x);
[XY1]  180:       x := x + 1;
[SS2]  181:       GotoXY(1,1);
[0R2]  182:       write(z);
[TU0]  183:     end;
[TU0]  184: end;
[000]  185:
[6S5]  186: procedure rechts;
[BH1]  187: begin
[MS8]  188:   if x > length(z) then write(BEL)
[DE3]  189:   else x := x + 1;
[TU0]  190: end;
[000]  191:
[BS5]  192: procedure links;
[BH1]  193: begin
[U36]  194:   if x = 1 then write(BEL)
[RE3]  195:   else x := x - 1;
[TU0]  196: end;
[000]  197:
[6A5]  198: procedure hoch;
[BH1]  199: begin
[KY5]  200:   if x < 78 then write(BEL)
[WH3]  201:   else x := x - 77;
[TU0]  202: end;
[000]  203:
[WU5]  204: procedure runter;
[BH1]  205: begin
[KK9]  206:   if x + 77 > length(z) then write(BEL)
[KH3]  207:   else x := x + 77;
[TU0]  208: end;
[000]  209:
[2C5]  210: procedure entf;
[BH1]  211: begin
[286]  212:   if x <= length(z) then
[BH1]  213:     begin
[9T3]  214:       delete(z, x, 1);
[SS2]  215:       GotoXY(1,1);
[V03]  216:       write(z,s);
[DL0]  217:     end
[GD4]  218:   else write(BEL);
```

```
[TU0]  219: end;
[000]  220:
[MM4]  221: procedure del;
[BH1]  222: begin
[U36]  223:   if x = 1 then write(BEL)
[A11]  224:   else
[BH1]  225:     begin
[4Y3]  226:       delete(z, x-1, 1);
[XY1]  227:       x := x - 1;
[SS2]  228:       GotoXY(1,1);
[V03]  229:       write(z,s);
[TU0]  230:     end;
[TU0]  231: end;
[000]  232:
[435]  233: procedure pos1;
[BH1]  234: begin
[XD1]  235:   x := 1;
[TU0]  236: end;
[000]  237:
[C95]  238: procedure ende;
[BH1]  239: begin
[R64]  240:   x := length(z) + 1;
[TU0]  241: end;
[000]  242:
[BH1]  243: begin
[H03]  244:   GotoXY(3,21);
[E2T]  245:   write(' Editieren Sie die Programmzeile
                                    [RETURN] Ende ');
          (***** geändert *****)
[Y14]  246:   window(2,6,78,10);
[NF2]  247:   ClrScr;
[0R2]  248:   write(z);
[SS2]  249:   GotoXY(1,1);
[XD1]  250:   x := 1;
[X52]  251:   repeat
[BU3]  252:     read(kbd, ch);
[RS2]  253:     case ch of
[WM0]  254:       ESC:
[BH1]  255:         begin
[BU3]  256:           read(kbd, ch);
[RS2]  257:           case ch of
[2Y2]  258:             'M': rechts;
[3H2]  259:             'K': links;
[B82]  260:             'H': hoch;
[313]  261:             'P': runter;
[282]  262:             'G': pos1;
[X72]  263:             'O': ende;
[682]  264:             'S': entf;
[TU0]  265:           end;
[TU0]  266:         end;
[WA2]  267:       BS: del;
[WT0]  268:       CR: ;
[D35]  269:       else schreibe(ch);
[TU0]  270:     end;
[PG7]  271:   GotoXY(x mod 77, x div 77 + 1);
[4R3]  272:   until ch = CR;
[TWE]  273:   if auto then z := copy(check, 2, 3) + SPA
          CE + z;  (***** NEU *****)
[TU0]  274: end;
[000]  275:
[000]  276:
[8V6]  277: procedure ZeigeZeile;
[BH1]  278: begin
[EP3]  279:   if Auto then
[BH1]  280:     begin
[XF8]  281:       check := '[' + copy(Zeile, 1, 3) + ']
          ';
[KSA]  282:       check[2] := UpCase(check[2]);  (*****
          NEU *****)
[VSA]  283:       check[3] := UpCase(check[3]);  (*****
          NEU *****)
[4TA]  284:       check[4] := UpCase(check[4]);  (*****
          NEU *****)
[CU6]  285:       Zeile := copy(Zeile, 5, 255);
[TU0]  286:     end;
[Y14]  287:   window(2,6,78,10);
[NF2]  288:   ClrScr;
[0G3]  289:   write(Zeile);
[R04]  290:   window(1,1,80,25);
[Y93]  291:   GotoXY(29,12);
[YY3]  292:   write(Nummer);
[TZ3]  293:   pruefe(Zeile);
[N93]  294:   GotoXY(10,12);
[YR2]  295:   write(ps);
[XV4]  296:   if bin then exit;
[EP3]  297:   if Auto then
[HT4]  298:     if ps = check then
[BH1]  299:       begin
[H03]  300:         GotoXY(3,21);
[CHC]  301:         write(' Prüfsumme OK für Zeile ',
          Nummer);
[PX4]  302:         weiter := true;
[DL0]  303:       end
[A11]  304:     else
[BH1]  305:       begin
[H03]  306:         GotoXY(3,21);
[WBS]  307:         write(BEL, ' Prüfsummenfehler !
          [E] Editieren   [RETURN] Weiter   [Ctrl-C]
          Abbrechen');
[U93]  308:         GotoXY(15,12);
[DL0]  309:       end
[A11]  310:   else
[BH1]  311:     begin
[H03]  312:       GotoXY(3,21);
```

```
[ZPM]  313:       write(' [E] Editieren      [RETURN] W
         eiter      [Q] Quitt      [Ctrl-C] Abbrechen
       ');
[U93]  314:       GotoXY(15,12);
[TU0]  315:     end;
[TU0]  316: end;
[000]  317:
[000]  318:
[XP5]  319: procedure Beenden;
[FK3]  320: var fi: file;
[BH1]  321: begin
[VE5]  322:   while not eof(t) do
[BH1]  323:     begin
[S25]  324:       readln(t, Zeile);
[V36]  325:       writeln(temp, Zeile);
[TU0]  326:     end;
[HY6]  327:   close(t); close(temp);
[CP5]  328:   if bin then close(f);
[PM6]  329:   assign(fi, Name + '.BAK');
[SY5]  330:   {$I-} reset(fi); {$I+}
[HK8]  331:   if IOresult = 0 then erase(fi);
[D16]  332:   rename(t, Name + '.BAK');
[HT6]  333:   rename(temp, Name + Ext);
[H03]  334:   GotoXY(3,21);
[DD6]  335:   if bin then Ext := '.BIN';
[7HF]  336:   write(' Die Datei ', Name, Ext, ' wurde g
       espeichert ');
[FR7]  337:   while WhereX < 78 do write(' ');
[X03]  338:   GotoXY(1,25);
[5B1]  339:   halt;
[TU0]  340: end;
[000]  341:
[000]  342:
[ES5]  343: procedure Quitt;
[BH1]  344: begin
[H03]  345:   GotoXY(3,21);
[VWK]  346:   write(' Drücken Sie [J], um die Arbeit zu
       beenden :                                  ');
[W93]  347:   GotoXY(48,21);
[WE3]  348:   read(kbd,ch);
[3N2]  349:   write(ch);
[3C7]  350:   if (ch = 'j') or (ch = 'J') then
[BH1]  351:     begin
[V36]  352:       writeln(temp, Zeile);
[7U2]  353:       Beenden;
[TU0]  354:     end;
[TU0]  355: end;
[000]  356:
[000]  357:
[BH1]  358: begin
[RB1]  359:   init;
[US2]  360:   if bin then
[BH1]  361:     begin
[H03]  362:       GotoXY(3,21);
[66C]  363:       write(' >>> Binärdatei wird erzeugt <
       <       ');
[TU0]  364:     end;
[VE5]  365:   while not eof(t) do
[BH1]  366:     begin
[S25]  367:       readln(t, Zeile);
[X52]  368:       repeat
[N45]  369:         weiter := false;
[U53]  370:         ZeigeZeile;
[867]  371:         if bin then weiter := true;
[Y45]  372:         if not weiter then
[BH1]  373:           begin
[BU3]  374:             read(kbd, ch);
[RS2]  375:             case ch of
[RK4]  376:               CR : weiter := true;
[7R4]  377:               'e','E': Edit(Zeile);
[HK9]  378:               'q','Q': if not Auto then Qui
       tt;
[TU0]  379:             end;
[TU0]  380:           end;
[XY3]  381:       until weiter;
[V36]  382:       writeln(temp, Zeile);
[US2]  383:       if bin then
[BH1]  384:         begin
[WLE]  385:           while copy(Zeile, 1, 1) = SPACE d
       o delete(Zeile, 1, 1);
[X52]  386:           repeat
[RH8]  387:             val('$' + copy(Zeile, 1, 2), b,
       code);
[RY8]  388:             if (code > 0) or (b < 0) or (b
       > 255) then
[BH1]  389:               begin
[H03]  390:                 GotoXY(3,21);
[CTD]  391:                 write(' +++++ ABBRUCH WEGEN
       DATENFEHLER +++++
                 ');
[X03]  392:                 GotoXY(1,25);
[5B1]  393:                 halt;
[TU0]  394:               end;
[VT1]  395:             by := b;
[LH3]  396:             write(f, by);
[0U6]  397:             Zeile := copy(Zeile, 3, 255);
[WLE]  398:             while copy(Zeile, 1, 1) = SPACE
       do delete(Zeile, 1, 1);
[PU4]  399:           until Zeile = EMPTY;
[TU0]  400:         end;
[6T5]  401:       Nummer := Nummer + 1;
[TU0]  402:     end;
[7U2]  403:   Beenden;
[5T0]  404: end.
```

**Listing 1. SCheck prüft jede Zeile auf Fehler**

```
[V16]    0:   VERSION = 'Version 2.3';
[H42]    1:   SPACE = #32;
[2G2]    2:   SPACE5 = '     ';
[Y3E]    3:   ZEICHEN : string[32] = '0123456789ABCDEFG
         HKLMNPRSTUVWXYZ';
[EMO]    4: var
[F1N]    5:   Nummern, Drucker, Elite, Schmal, Fett, Tx
         t, Pruef, HexDump: boolean;
[FEL]    6:   Summe, Breite, Hoehe, ZeilenNummer, Druck
         Zeile, Zaehler: integer;
[RG2]    7:   p, t: text;
[MN2]    8:   f: BinFile;
[FB2]    9:   LZ: char;
[PM2]   10:   b: byte;
[YB5]   11: procedure info;
[BH1]   12: begin
[9Z2]   13:   writeln;
[EVD]   14:   writeln(BEL, '+++++ Fehler bei Parametera
         ngabe !');
```

**Listing 2. So werden Pascal-Programme ab sofort gedruckt**

```
  VERSION = 'Version 2.3';
  SPACE = #32;
  SPACE5 = '     ';
  ZEICHEN : string[32] = '0123456789ABCDEFGHKLMNPRSTUVWXYZ';
var
  Nummern, Drucker, Elite, Schmal, Fett, Txt, Pruef, HexDump: boolean;
  Summe, Breite, Hoehe, ZeilenNummer, DruckZeile, Zaehler: integer;
  p, t: text;
  f: BinFile;
  LZ: char;
  b: byte;
procedure info;
begin
  writeln;
  writeln(BEL, '+++++ Fehler bei Parameterangabe !');
```

**Listing 3. So müssen Sie das Pascal-Listing eingeben**

```
V16   VERSION = 'Version 2.3';
H42   SPACE = #32;
2G2   SPACE5 = '     ';
Y3E   ZEICHEN : string[32] = '0123456789ABCDEFGHKLMNPRSTUVWXYZ';
EMO var
F1N   Nummern, Drucker, Elite, Schmal, Fett, Txt, Pruef, HexDump: bool
FEL   Summe, Breite, Hoehe, ZeilenNummer, DruckZeile, Zaehler: integer
RG2   p, t: text;
MN2   f: BinFile;
FB2   LZ: char;
PM2   b: byte;
YB5 procedure info;
BH1 begin
9Z2   writeln;
EVD   writeln(BEL, '+++++ Fehler bei Parameterangabe !');
```

**Listing 4. Pascal-Programme mit Prüfsumme eingegeben**

```
[PRC] 1010 SCREEN 9:KEY OFF:CLS:COLOR 1,0:PALETTE 1,7:P
      ALETTE 7,1
[6U4] 1020 COLORSW=1:GOTO 1100
[213] 1025 RESUME 1030
[H3B] 1030 ON ERROR GOTO 0:SCREEN 2:KEY OFF:CLS:COLORSW
      =0
[164] 1040 VY1=199:VY2=150
[CDK] 1100 LOCATE 10,1:PRINT "Please type the longitude
       of your country"
```

**Listing 5. Basic-Programme sehen ab sofort in der DOS so aus**

```
1010 SCREEN 9:KEY OFF:CLS:COLOR 1,0:PALETTE 1,7:PALETTE 7,1
1020 COLORSW=1:GOTO 1100
1025 RESUME 1030
1030 ON ERROR GOTO 0:SCREEN 2:KEY OFF:CLS:COLORSW=0
1040 VY1=199:VY2=150
1100 LOCATE 10,1:PRINT "Please type the longitude of your country"
```

**Listing 6. Bei der Eingabe müssen Sie auf die Zeilennummer in eckigen Klammern verzichten**

```
PRC 1010 SCREEN 9:KEY OFF:CLS:COLOR 1,0:PALETTE 1,7:PALETTE 7,1
6U4 1020 COLORSW=1:GOTO 1100
213 1025 RESUME 1030
H3B 1030 ON ERROR GOTO 0:SCREEN 2:KEY OFF:CLS:COLORSW=0
164 1040 VY1=199:VY2=150
CDK 1100 LOCATE 10,1:PRINT "Please type the longitude of your country"
```

**Listing 7. Auch Basic-Programme können mit Prüfsumme eingegeben werden**

# Generaluntersuchung mit dem Festplattendoktor

**In vielen Fällen – beispielsweise bei beschädigten Dateien – ist es sehr hilfreich, sich den Inhalt einer Diskette oder Festplatte ansehen und manipulieren zu können. Zu diesem Zweck nimmt der Festplattendoktor Ihren Datenträger genau unter die Lupe.**

Viele Anwender stehen vor der Situation, daß Ihre Datenträger kränkeln oder nicht mehr das beinhalten, was sie eigentlich sollten. In diesen Fällen sollten Sie den Festplattendoktor konsultieren. Er weiß fast immer Rat und hilft Ihnen, den Datenträger wieder auf die Beine zu bringen.

Der Festplattendoktor (Listing) ist in Turbo Pascal 3.0 geschrieben und enthält alle wesentlichen Funktionen, die solch ein Editor benötigt. Er zeigt nach dem Start im unteren Bereich des Bildschirms die erlaubten Befehle an. Der obere Bereich dient zur Ausgabe der Datenträger. Die Befehle des Festplattendoktors werden mit den Funktionstasten ausgewählt. Die Tabelle zeigt in einer Übersicht alle Programmfunktionen.

Der Festplattendoktor zeigt auf Wunsch den Inhalt eines Sektors einer Festplatte oder Diskette in Form von ASCII-Zeichen oder in hexadezimaler Notation an. Mit den Pfeiltasten führen Sie den Cursor über das gesamte Datenfeld. Sind Sie an der Position angekommen, an der Sie Daten ändern wollen, dann geben Sie einfach die neuen Werte ein.

Der Sektor einer Diskette enthält bekanntlich 512 Byte. Weil der Festplattendoktor aber nur 256 Byte gleichzeitig anzeigen kann, wird jeder Sektor in zwei Bildschirmseiten aufgeteilt. Mit der PgDn-Taste schalten Sie zwischen diesen beiden Seiten um.

Mit [F2] wählen Sie das Laufwerk aus, um das sich der Diskettendoktor kümmern soll. Geben Sie zu diesem Zweck den gewünschten Laufwerksbuchstaben ein (beispielsweise »B«) und drücken Sie die Enter-Taste. Ähnlich verfahren Sie mit der Funktionstaste [F1], die dazu dient, einen bestimmten Sektor anzuwählen. Der Unterschied besteht darin, daß Sie in diesem Fall eine Zahl eingeben müssen, die den zu untersuchenden Sektor angibt.

## Über die Funktionstasten wählen Sie die verschiedenen Menüpunkte des Festplattendoktors aus

Mit [F3] und [F4] lesen Sie den folgenden beziehungsweise vorherigen Sektor von der Diskette. Die Funktionstaste [F8] schreibt den aktuellen Sektor auf den Datenträger.

Mit dem Festplattendoktor bereitet es Ihnen keine Schwierigkeit, den Inhalt einzelner Disketten übersichtlich darzustellen und einfach zu verändern. Doch neben Diskettenlaufwerken bearbeitet der Doktor auch Festplatten, denn das Betriebssystem MS-DOS steuert eine Festplatte im Prinzip genauso an wie ein Diskettenlaufwerk.

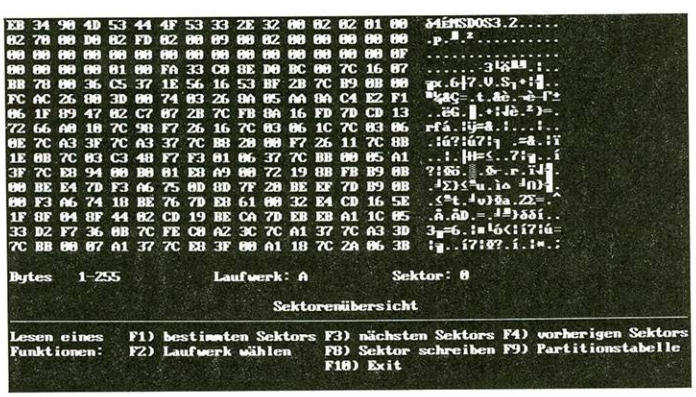
Doch obwohl der Anwender auch bei der Festplatte die Spur und den Sektor spezifizieren muß, gibt es hier einige Unterschiede. So kann der Benutzer eine Festplatte für verschiedene Betriebssysteme einrichten. Man kann beispielsweise einen Teil der Platte für MS-DOS, einen weiteren Teil für Xenix und einen anderen für PC-MOS reservieren. Dazu wird die Festplatte in mehrere Bereiche aufgeteilt, die als »Partitions« (englisch für »Aufteilung«) bezeichnet werden.

Weil die Partition eine wichtige Rolle bei der Verwaltung der Daten des Anwenders spielt, informiert Sie der Festplattendoktor über die wichtigsten zugehörigen Daten. Die Informationen, welches Betriebssystem welche Bereiche der Festplatte benutzt, sind in der sogenannten »Partition Table« (englisch für »Aufteilungstabelle«) abgelegt. In dieser Tabelle ist gespeichert, an welcher Stelle welches Betriebssystem abgelegt ist, wie lang jeder Bereich ist und welcher Bereich aktiviert wurde.

Wenn Sie die Funktionstaste [F9] drücken und zuvor mit [F2] eine Festplatte angewählt haben, zeigt Ihnen der Festplattendoktor die Partitionstabelle dieses Speichermediums an. Das Programm liest diese Daten über den Datenträger-Interrupt 13hex ein. Die Partitionstabelle ist 64 Byte lang und befindet sich im ersten Sektor auf der ersten Spur des ersten Kopfes. Der Festplattendoktor liest die Tabelle in den Datenpuffer der Feldvariablen »datenfeld« ein.

## Auf einer Festplatte lassen sich unterschiedliche Betriebssysteme gemeinsam verwalten

Weil jeder Informationsblock aus 16 Byte besteht, lassen sich höchstens vier Betriebssysteme gleichzeitig einsetzen. Der Festplattendoktor zeigt Ihnen aus diesem Grund vier Felder auf dem Bildschirm an, die Informationen über die eingeteilten Bereiche geben. In der obersten Zeile jedes Blocks gibt er an, ob die zugehörige Partition aktiv ist. Die nächste Zeile gibt darüber Aufschluß, ob der Bereich zu MS-DOS gehört oder mit einem unbekannten Betriebssystem be-




**Mit dem Festplattendoktor lassen sich Datenträger lesen und editieren**

| Taste | Funktion |
|---|---|
| [F1] | Lesen eines bestimmten Sektors in den Datenpuffer |
| [F2] | Auswählen des aktiven Laufwerks |
| 0[F3] | Lesen des folgenden Sektors in den Datenpuffer |
| [F4] | Lesen des vorherigen Sektors in den Datenpuffer |
| [F8] | Schreiben des aktuellen Sektors auf den Datenträger |
| [F9] | Anzeige der Partitionstabelle der Festplatte |
| [F10] | Beenden des Festplattendoktors |

**Tabelle. Diese Kommandos versteht der Festplattendoktor**

legt ist. Die letzten beiden Zeilen informieren Sie über den physikalischen Anfang und das Ende der jeweiligen Partition. Nach der Anzeige der Partitionstabelle kehren Sie durch das Betätigen einer beliebigen Taste in das Hauptmenü zurück.

Der Festplattendoktor hat gegenüber vielen anderen Diskketteneditoren einen Vorteil: Fast alle PC-Betriebssysteme benutzen die gleichen BIOS-Interrupts für das interne Datenmanagement. Über das BIOS können Sie also auf die gesamte Festplatte zugreifen, während die Aktivitäten von MS-DOS auf die Bereiche des Datenträgers begrenzt sind, die auch für MS-DOS eingerichtet wurden. Weil der Festplattendoktor nun die BIOS-Interrupts benutzt, kann er problemlos unterschiedliche Betriebssysteme auf einer Festplatte zur gleichen Zeit bearbeiten.

Die Funktionstaste [F10] ist der letzte Befehl des Festplattendoktors und dient zum Beenden des Programms.

(Markus Zietlow/ma)

```
[KA6]    0: program Diskdoktor;
[000]    1:
[X0B]    2: (* Programmautor: Markus Zietlow
[ML9]    3:    (C)opyright:   DOS International
[YB9]    4:    Funktion:      Diskketteneditor *)
[000]    5:
[VM1]    6: const
[FR2]    7:   Fehler=-1;
[000]    8:
[A21]    9: type
[AA6]   10:   Registersatz = record
[7KC]   11:     ax,bx,cx,dx,bp,si,di,ds,es,flags:intege
          r;
[TU0]   12:     end;
[000]   13:
[HK7]   14:   Feld = array[0..5000] of byte;
[000]   15:
[EM0]   16: var
[RZ5]   17:   Hauptlaufwerk:byte;
[Y75]   18:   Sektor:integer;
[MC3]   19:   Puffer:Feld;
[KU7]   20:   Register:Registersatz;
[000]   21:
[7RK]   22: function Sektor_lesen(Laufwerk:byte; Sektr,
          Anzahl:integer;
[TZ7]   23:                           var Puffer:Feld):inte
          ger;
[000]   24:
[BH1]   25:   begin
[VFG]   26:     inline ($55/$8b/$ec/$83/$c5/$08/$1e/$8b
          /$5e/$00/$8b/$46/$02/$8e/$d8/$8b/$4e/$04/
[2EC]   27:                 $8b/$56/$06/$8a/$46/$08/$55/$cd
          /$25/$73/$08/$9d/$5d/$89/$46/$0a/
[0H9]   28:                 $eb/$08/$90/$9d/$5d/$33/$c0/$89
          /$46/$0a/$1f/$5d);
[TU0]   29:   end;
[000]   30:
[A9N]   31: function Sektor_schreiben(Laufwerk:byte; Se
          ktr,Anzahl:integer;
[TZ7]   32:                                 var Puffer:Feld):
          integer;
[000]   33:
[BH1]   34:   begin
[VFG]   35:     inline ($55/$8b/$ec/$83/$c5/$08/$1e/$8b
          /$5e/$00/$8b/$46/$02/$8e/$d8/$8b/$4e/$04/
[5EC]   36:                 $8b/$56/$06/$8a/$46/$08/$55/$cd
          /$26/$73/$08/$9d/$5d/$89/$46/$0a/
[0H9]   37:                 $eb/$08/$90/$9d/$5d/$33/$c0/$89
          /$46/$0a/$1f/$5d);
[TU0]   38:   end;
[000]   39:
[XX5]   40: procedure Warnung;
[BH1]   41:   begin
[RH3]   42:     write (chr(7));
[TU0]   43:   end;
[000]   44:
[5F6]   45: procedure Titelbild;
[BH1]   46:   begin
[R04]   47:     window(1,1,80,25);
[NM2]   48:     clrscr;
[CC3]   49:     gotoxy(33,1);
[NK6]   50:     writeln('===============');
[FC3]   51:     gotoxy(33,2);
[B39]   52:     writeln('Diskettendoktor');
[KC3]   53:     gotoxy(33,3);
[NK6]   54:     writeln('===============');
[4D3]   55:     gotoxy(23,9);
[DUD]   56:     writeln('(C)opyright 1988: DOS Internat
          ional');
[AP3]   57:     gotoxy(29,15);
[UMA]   58:     writeln('Autor: Markus Zietlow');
[UC3]   59:     gotoxy(6,24);
[12S]   60:     writeln('Drücken Sie bitte eine Taste,
          um den Diskettendoktor zu starten !');
[TU0]   61:   end;
[000]   62:
[TTB]   63: function Hexadezimal(Zahl:byte):char;
[BH1]   64:   begin
[FE8]   65:     if (Zahl<=0) or (Zahl>15) then
[CL4]   66:       Hexadezimal:='0'
[A11]   67:     else
[3N8]   68:       if (Zahl>0) and (Zahl<=9) then
[PA9]   69:         Hexadezimal:=chr(ord('1')+Zahl-1)
[A11]   70:       else
[GP9]   71:         Hexadezimal:=chr(ord('A')+Zahl-10);
[TU0]   72:   end;
[000]   73:
[DCB]   74: function Dezimal(letter:char):byte;
[BH1]   75:   begin
[087]   76:     if letter in ['0'..'9'] then
[FE8]   77:       Dezimal:=ord(letter)-ord('0')
[A11]   78:     else
[1A7]   79:       if letter in ['A'..'F'] then
[V29]   80:         Dezimal:=ord(letter)-ord('A')+10
[A11]   81:       else
[1F7]   82:         if letter in ['a'..'f'] then
[V59]   83:           Dezimal:=ord(letter)-ord('a')+10
[A11]   84:         else
[BN5]   85:           Dezimal:=Fehler;
[TU0]   86:   end;
[000]   87:
[AAE]   88: procedure Laufwerk_holen(var Datenfeld:Feld
          );
[EM0]   89:   var
[SX7]   90:     Laufwerk_neu:string[1];
[MZ6]   91:     Laufwerk_alt:integer;
[BH1]   92:   begin
[R04]   93:     window(1,1,80,25);
[KP3]   94:     gotoxy(35,18);
[BH2]   95:     write ('?');
[KP3]   96:     gotoxy(35,18);
[MN5]   97:     read  (Laufwerk_neu);
[K28]   98:     if ord(Laufwerk_neu)>96 then
[XVB]   99:       Laufwerk_neu:=chr(ord(Laufwerk_neu)-3
          2);
[NY8]  100:     Laufwerk_alt:=Hauptlaufwerk;
[N9B]  101:     Hauptlaufwerk:=ord(Laufwerk_neu)-65;
[9EH]  102:     if Sektor_lesen(Hauptlaufwerk,Sektor,1,
          Datenfeld) <> 0 then
[BH1]  103:     begin
[NY2]  104:       Warnung;
[H39]  105:       Hauptlaufwerk:=Laufwerk_alt;
[TU0]  106:     end;
[TU0]  107:   end;
[000]  108:
[ZDD]  109: procedure Neuer_Sektor(var Datenfeld:Feld);
[EM0]  110:   var
[W67]  111:     Sektor_neu:string[20];
[RY8]  112:     Sektor_alt,Korrekt:integer;
[BH1]  113:   begin
[R04]  114:     window(1,1,80,25);
[HP3]  115:     gotoxy(54,18);
[A83]  116:     write('?         ');
[HP3]  117:     gotoxy(54,18);
[8X4]  118:     read  (Sektor_neu);
[2W5]  119:     Sektor_alt:=Sektor;
[3P9]  120:     val(Sektor_neu,Sektor,Korrekt);
[KY4]  121:     if Korrekt<>0 then
[BH1]  122:     begin
[NY2]  123:       Warnung;
[Y16]  124:       Sektor:=Sektor_alt;
[TU0]  125:     end;
[HUH]  126:     if Sektor_lesen(Hauptlaufwerk,Sektor,1,
          Datenfeld)<>0 then
[BH1]  127:     begin
[NY2]  128:       Warnung;
[Y16]  129:       Sektor:=Sektor_alt;
[TU0]  130:     end;
[TU0]  131:   end;
[000]  132:
[PAC]  133: function Hex_Feld(X,Y:integer):boolean;
[BH1]  134:   begin
[NGG]  135:     if (X>=1) and (X div 3<16) and (Y<=17)
          and (Y>=1) and (X mod 3<>0) then
[5T4]  136:       Hex_Feld:=true
[A11]  137:     else
[1D5]  138:       Hex_Feld:=false
[TU0]  139:   end;
[000]  140:
[VWC]  141: function ASCII_Feld(X,Y:integer):boolean;
[BH1]  142:   begin
[9TA]  143:     if (X>=50) and (X<=65) and (Y<=17) and
          (Y>=1) then
[YU4]  144:       ASCII_Feld:=true
[A11]  145:     else
[EU4]  146:       ASCII_Feld:=false;
[TU0]  147:   end;
[000]  148:
```

```
[LYB]  149: procedure Statuszeile(Seite:integer);
[BH1]  150:   begin
[Y54]  151:     window(1,18,80,19);
[NM2]  152:     clrscr;
[H55]  153:     if Seite=0 then
[TN4]  154:       write('Bytes   1-255')
[A11]  155:     else
[3A5]  156:       write('Bytes 256-512');
[XSM]  157:     writeln('           Laufwerk: ',chr(65+
          Hauptlaufwerk),'          Sektor: ',Sektor);
[TU0]  158:   end;
[000]  159:
[EDG]  160: procedure Ausgabe(Reihe,Spalte:integer;valu
          e:byte);
[BH1]  161:   begin
[V26]  162:     gotoxy(Spalte,Reihe);
[5M9]  163:     write(Hexadezimal(value shr 4));
[EX9]  164:     write(Hexadezimal(value and $f));
[PS8]  165:     gotoxy((Spalte div 3)+50,Reihe);
[W15]  166:     if value<32 then
[T62]  167:       write('.')
[A11]  168:     else
[8F5]  169:       write(chr(value));
[TU0]  170:   end;
[000]  171:
[XGG]  172: procedure Zeige_Seite(Datenfeld:Feld;Seite:
          integer);
[EM0]  173:   var
[HA6]  174:     Reihe,Spalte:integer;
[7E3]  175:     Inhalt:byte;
[BH1]  176:   begin
[G04]  177:     window(1,1,80,17);
[NM2]  178:     clrscr;
[SU5]  179:     for Reihe:=0 to 15 do
[BZ5]  180:       for Spalte:=0 to 15 do
[BH1]  181:       begin
[R3D]  182:         Inhalt:=Datenfeld[Seite+Reihe*16+Sp
          alte];
[Z8A]  183:         Ausgabe(Reihe+1,3*Spalte+1,Inhalt);
[TU0]  184:       end;
[TU0]  185:   end;
[000]  186:
[H6K]  187: procedure Hexadezimalausgabe(var X,Y,Parame
          ter:integer;
[YEA]  188:                                 var Datenfeld:
          Feld;Seite:integer);
[EM0]  189:   var
[1F7]  190:     Position,Wert:integer;
[BH1]  191:   begin
[926]  192:     if Hex_Feld(X,Y) then
[BH1]  193:     begin
[497]  194:       Position:=(X-1) div 3 + (Y-1)*16;
[U3B]  195:       Wert:=Dezimal(chr(Parameter and $ff))
          ;
[YG6]  196:       if Wert<>Fehler then
[BH1]  197:       begin
[BE4]  198:         if X mod 3=1 then
[WRM]  199:           Datenfeld[Position+Seite]:=(Daten
          feld[Position+Seite] and $f)+Wert*16
[A11]  200:         else
[6EM]  201:           Datenfeld[Position+Seite]:=(Daten
          feld[Position+Seite] and $f0)+Wert;
[LXP]  202:         Ausgabe(1+Position div 16,(Positi
          on mod 16)*3+1,Datenfeld[Position+Seite]);
[XL1]  203:         X:=X+1;
[TU0]  204:       end;
[DL0]  205:     end
[A11]  206:     else
[2T6]  207:       if ASCII_Feld(X,Y) then
[BH1]  208:       begin
[GS5]  209:         Position:=X-50+(Y-1)*16;
[2HE]  210:         Datenfeld[Position+Seite]:=Paramete
          r and $ff;
[LXP]  211:         Ausgabe(1+Position div 16,(Position
           mod 16)*3+1,Datenfeld[Position+Seite]);
[XL1]  212:         X:=X+1;
[TU0]  213:       end;
[TU0]  214:   end;
[000]  215:
[8WK]  216: procedure Zeige_Inhalt(Datenfeld:Feld;Seite
          ,Modus:integer);
[BH1]  217:   begin
[S55]  218:     if Modus=2 then
[869]  219:       Zeige_Seite(Datenfeld,Seite*256);
[TU0]  220:   end;
[000]  221:
[GE6]  222: procedure Kommandos;
[BH1]  223:   begin
[354]  224:     window(1,20,80,24);
[NM2]  225:     clrscr;
[7WC]  226:     writeln('
          Sektorenübersicht');
[LSE]  227:     write  ('------------------------------
          ----------------------------------------
          ------');
[V7U]  228:     write  ('Lesen eines   F1) bestimmten S
          ektors F3) nächsten Sektors F4) vorherigen S
          ektors');
[HXU]  229:     writeln('Funktionen:   F2) Laufwerk wäh
          len    F8) Sektor schreiben F9) Partitionsta
          belle');
[Z89]  230:     write  ('
            F10) Exit');
[TU0]  231:   end;
[000]  232:
[0VD]  233: procedure Zeige_Partition(Datenfeld:Feld);
[VM1]  234:   const
[VA3]  235:     Zeiger=$1be;
[EM0]  236:   var
[RT5]  237:     Zaehler:integer;
[KU7]  238:     Register:Registersatz;
[BH1]  239:   begin
[XR4]  240:     Register.ax:=$201;
[1P4]  241:     Register.dx:=$80;
[7R4]  242:     Register.cx:=1;
[3T8]  243:     Register.es:=seg(Datenfeld);
[VS8]  244:     Register.bx:=ofs(Datenfeld);
[V25]  245:     intr($13,Register);
[R04]  246:     window(1,1,80,25);
[NM2]  247:     clrscr;
[7WD]  248:     writeln('                                F
          estplatten-Partition');
[NXL]  249:     writeln('==============================
          ==========================================
          =====');
[0R5]  250:     for Zaehler:=0 to 3 do
[BH1]  251:     begin
[BSC]  252:       gotoxy((Zaehler mod 2)*40+1,(Zaehler
          div 2)*8+6);
[H79]  253:       write('Partition ',Zaehler+1,': ');
[HTB]  254:       if Datenfeld[Zeiger+Zaehler*16]<>$80
          then
[GK3]  255:         write('Nicht')
[A11]  256:       else
[VZ2]  257:         write('Ist');
[KX5]  258:       writeln(' aktiv');
[TSC]  259:       gotoxy((Zaehler mod 2)*40+1,(Zaehler
          div 2)*8+8);
[LC5]  260:       write('DOS-Part.: ');
[8NB]  261:       if Datenfeld[Zeiger+Zaehler*16+4]=1 t
          hen
[FY9]  262:         write('12-Bit File Allocation Table
          ')
[A11]  263:       else
[HNB]  264:         if Datenfeld[Zeiger+Zaehler*16+4]=4
           then
[LY9]  265:           write('16-Bit File Allocation Tab
          le')
[A11]  266:         else
[BY3]  267:           write('Keine');
[0TC]  268:       gotoxy((Zaehler mod 2)*40+1,(Zaehler
          div 2)*8+9);
[YB0]  269:       writeln('Anfang: Kopf ',Datenfeld[Zei
          ger+Zaehler*16+1],' Sektor ',Datenfeld[Zeige
          r+Zaehler*16+2],' Spur ',
[8U9]  270:                 Datenfeld[Zeiger+Zaehler*16+
          3]);
[CUC]  271:       gotoxy((Zaehler mod 2)*40+1,(Zaehler
          div 2)*8+10);
[TMZ]  272:       writeln('Ende:   Kopf ',Datenfeld[Zei
          ger+Zaehler*16+5],' Sektor ',Datenfeld[Zeige
          r+Zaehler*16+6],' Spur ',
[WU9]  273:                 Datenfeld[Zeiger+Zaehler*16+
          7]);
[TU0]  274:     end;
[MC3]  275:     gotoxy(8,22);
[YUR]  276:     writeln('Drücken Sie bitte eine Taste,
          um in den Editor zurückzukehren !');
[XP4]  277:     Register.ax:=0;
[V25]  278:     intr($16,Register);
[TU0]  279:   end;
[000]  280:
[X8E]  281: procedure Zeige_Sektoren(var Datenfeld:Feld
          );
[EM0]  282:   var
[ZW5]  283:     Schluss:boolean;
[G38]  284:     Seite,X,Y,Modus:integer;
[KU7]  285:     Register:Registersatz;
[BH1]  286:   begin
[RH5]  287:     Schluss:=false;
[951]  288:     X:=50;
[FV0]  289:     Y:=1;
[3B2]  290:     Seite:=0;
[CD2]  291:     Modus:=2;
[HUH]  292:     if Sektor_lesen(Hauptlaufwerk,Sektor,1,
          Datenfeld)<>0 then

[NY2]  293:       Warnung;
[223]  294:     Kommandos;
[LY5]  295:     Statuszeile(Seite);
[YF8]  296:     Zeige_Seite(Datenfeld,Seite);
[446]  297:     while not Schluss do
[BH1]  298:     begin
[873]  299:       gotoxy(X,Y);
[XP4]  300:       Register.ax:=0;
[V25]  301:       intr($16,Register);
[D98]  302:       if Register.ax and $ff=0 then
[DS5]  303:       case (Register.ax) of
[2E2]  304:         $3b00: begin
[DE7]  305:                  Neuer_Sektor(Datenfeld);
[223]  306:                  Kommandos;
[LY5]  307:                  Statuszeile(Seite);
```

```
[XDA]  308:                    Zeige_Inhalt(Datenfeld,Sei
       te,Modus);
[TU0]  309:                  end;
[5E2]  310:          $3c00: begin
[4E8]  311:                    Laufwerk_holen(Datenfeld);
[223]  312:                    Kommandos;
[LY5]  313:                    Statuszeile(Seite);
[XDA]  314:                    Zeige_Inhalt(Datenfeld,Sei
       te,Modus);
[TU0]  315:                  end;
[8E2]  316:          $3d00: begin
[885]  317:                    Sektor:=Sektor+1;
[HUH]  318:                    if Sektor_lesen(Hauptlaufw
       erk,Sektor,1,Datenfeld)<>0 then
[BH1]  319:                    begin
[NY2]  320:                      Warnung;
[P85]  321:                      Sektor:=Sektor-1;
[TU0]  322:                    end;
[LY5]  323:                    Statuszeile(Seite);
[XDA]  324:                    Zeige_Inhalt(Datenfeld,Seit
       e,Modus);
[TU0]  325:                  end;
[6R6]  326:          $3e00: if Sektor>0 then
[BH1]  327:                  begin
[P85]  328:                    Sektor:=Sektor-1;
[SRM]  329:                    if Sektor_lesen(Hauptlaufw
       erk,Sektor,1,Datenfeld)<>0 then Warnung;
[LY5]  330:                    Statuszeile(Seite);
[XDA]  331:                    Zeige_Inhalt(Datenfeld,Sei
       te,Modus);
[DL0]  332:                  end
[A11]  333:                  else
[NY2]  334:                    Warnung;
[2HR]  335:          $4200: if Sektor_schreiben(Hauptlau
       fwerk,Sektor,1,Datenfeld)<>0 then Warnung;
[R92]  336:          $4300: begin
[PS8]  337:                    Zeige_Partition(Datenfeld)
       ;
[NM2]  338:                    clrscr;
[223]  339:                    Kommandos;
[LY5]  340:                    Statuszeile(Seite);
```

```
[XDA]  341:                    Zeige_Inhalt(Datenfeld,Sei
       te,Modus);
[TU0]  342:                  end;
[SA5]  343:          $4400: Schluss:=true;
[BF3]  344:          $4800: if Y>1 then
[CM1]  345:                   Y:=Y-1;
[6L3]  346:          $4b00: if X>1 then
[7M1]  347:                   X:=X-1;
[AF3]  348:          $4d00: if X<65 then
[XL1]  349:                   X:=X+1;
[0A3]  350:          $5000: if Y<16 then
[2M1]  351:                   Y:=Y+1;
[L92]  352:          $5100: begin
[276]  353:                   Seite:=(Seite+1) mod 2;
[LY5]  354:                   Statuszeile(Seite);
[XDA]  355:                   Zeige_Inhalt(Datenfeld,Sei
       te,Modus);
[TU0]  356:                  end;
[DL0]  357:        end
[A11]  358:        else
[S55]  359:        if Modus=2 then
[R2H]  360:          Hexadezimalausgabe(X,Y,Register.ax,
       Datenfeld,Seite*256);
[TU0]  361:      end;
[TU0]  362:    end;
[000]  363:
[BH1]  364: begin
[YXF]  365:    Hauptlaufwerk:=0;          (* Beginn des H
       auptprogramms *)
[1U2]  366:    Sektor:=0;
[R13]  367:    Titelbild;
[XP4]  368:    Register.ax:=0;
[V25]  369:    intr($16,Register);
[MA7]  370:    Zeige_Sektoren(Puffer);
[R04]  371:    window(1,1,80,25);
[NM2]  372:    clrscr;
[5T0]  373: end.
```

**Listing. Mit dem Festplattendoktor bringen Sie Ihren Datenträger wieder auf Vordermann**


# TSENG LABS EVA 1024

## DER NEUE VGA STANDARD

**100% Register-Level Kompatibilität zu IBM VGA/EGA Standard**
**Volle Kompatibilität zu OS/2 und WINDOWS 386**

- **Abwärts kompatibel zu MDA, CGA, MCGA und Hercules**
- **Darstellung von 64 Graustufen in allen Monochrom-Modi**
- **Hochauflösende Grafik 640x480 mit 256 aus 262144 Farben**
- **Hochauflösende Grafik 800x600, 1024x768 mit 16 aus 262144 Farben**
- **Hardware Features: Window, Zoom, Pan**
- **Arbeitet mit TTL-, Analog- und Multifrequenz-Monitoren, analog und digital**
- **bis zu 8 simultan ladbare Schriftarten**
- **stellt 132 Spalten in 60, 44, 32, 29 und 25 Zeilen dar**
- **Videofrequenz 65 MHz, absolut flimmerfreies Bild**

Überall im guten Fachhandel erhältlich

KOK KULKONI

Kulenkampff & Konitzky, Kohlhökerstraße 19, 2800 Bremen 1
Tel. 0421/3676-278, Fax 0421/367 62 48, Telex 244 248 kuko d

bäumer foto + design

# Optimierung mit System – »config.sys«

**Neben der Stapeldatei »autoexec.bat« ist »config.sys« die wichtigste Datei Ihres Betriebssystems. Mit diesen Angaben konfiguren Sie bei jedem Start Ihr System. Worauf Sie bei Gebrauch der wichtigsten Befehle achten müssen, erfahren Sie im folgenden Bericht.**

Nachdem Sie Ihren PC eingeschaltet oder einen Neustart mit dem Reset-Schalter oder der Tastenkombination [Alt-Ctrl-Del] durchgeführt haben, lädt der Computer das Betriebssystem. Dazu gehören die zwei versteckten Dateien »msdos.sys« und »io.sys«. Danach folgt schon die Datei »config.sys« – noch vor »command.com« und der Datei »autoexec .bat«. Mit dieser Config-Datei teilen Sie dem Betriebssystem Änderungen der Systemeinstellung mit. Dazu gehören unter anderem zusätzliche Hardwarekomponenten oder spezielle Treiber-Dateien.

Die Datei befindet sich im Hauptverzeichnis Ihrer Bootdiskette, und ist nicht in einem Unterverzeichnis versteckt. Diese Datei läßt sich mit jedem beliebigen Texteditor, beispielsweise Edlin von MS-DOS oder dem Befehl

```
copy con config.sys
```

anlegen.

Entscheiden Sie sich für den Copy-Befehl, haben Sie folgendes zu beachten: Ihre alte »config.sys« wird überschrieben und ist somit ungültig! Beenden Sie Ihre Eingabe mit [F6] und MS-DOS kopiert die Datei auf Ihren Datenträger. Doch nun zu den einzelnen Parametern beziehungsweise Befehlen, die in der »config.sys« festgelegt werden.

Der Befehl »break« kennt zwei Parameter: »on« und »off«. Tragen Sie also in Ihrer »config.sys«

```
break = on
```

ein, können Sie mit der Tastenkombination [Ctrl-C] Diskettenzugriffe abbrechen. Diese Einstellung empfiehlt sich besonders dann, wenn Sie mit Programmen arbeiten, die sehr oft auf die Diskette oder Festplatte zurückgreifen. Bei Fehlern bereitet es dann keine Schwierigkeiten, das Programm beziehungsweise den Zugriff abzubrechen, ohne das System neu zu starten. Der Befehl »break« kann aber auch außerhalb der »config.sys« benutzt werden. Wenn Sie auf der Betriebssystemebene

```
break
```

eingeben, teilt Ihnen MS-DOS mit, ob der Befehl aktiv beziehungsweise »on« ist, oder auf »off« steht. Der Zustand von »break« läßt sich nun jederzeit durch die Eingabe

```
break on
```

oder

```
break off
```

verändern.

Eine Anweisung, die nach dem Eintrag in die Config-Dateien unveränderlich feststeht und nachträglich nicht mehr zu ändern ist, heißt »buffers«. Mit der Zeile

```
buffers = 15
```

legen Sie einen Puffer von 7,5 KByte im Hauptspeicher für Disketten beziehungsweise Festplattenzugriffe an. Arbeiten Sie mit großen Dateien oder mit Anwendung, die viele Overlays anlegen, wird immer zuerst auf den Puffer und dann erst auf den Massenspeicher zugegriffen. Je mehr Daten sich nun bereits in diesem Zwischenspeicher befinden, desto schneller werden die Zugriffe, denn der Computer findet die Information bereits im Puffer und der Sprung zur Diskette entfällt. Um so kürzer sind demnach auch Ihre Wartezeiten. Der Nachteil an großen Puffern ist jedoch, daß sie wichtigen Arbeitsspeicher belegen.

## Mit »country« legen Sie die Form von Datums- und Dezimalschreibweise fest

Mit dem Parameter »country« teilen Sie MS-DOS mit, in welcher Form Datum, Uhrzeit, Währungssymbol und Zahlenangaben erfolgen sollen. Übergehen Sie diese Angaben, so erscheint das Datum nicht in der Reihenfolge Tag-Monat-Jahr, sondern als Monat-Tag-Jahr. Auch akzeptiert der Computer bei Zahlenangaben, anstatt des bei uns beispielsweise üblichen Kommas in Preisangaben, dann nur die spezifische amerikanische Schreibweise mit einem Punkt. Diese Angaben werden dann auch von Ihrer jeweiligen Applikation berücksichtigt. Achten also darauf, daß Ihr »country« stimmt. Für Deutschland tragen Sie

```
country = 049
```

ein. Diese Zahlenkombination wurde der Einfachheit halber von den Auslands-Telefonvorwahlnummern übernommen (für die Bundesrepublik steht 0049). Dieser Befehl beeinflußt jedoch die Tastaturbelegung nicht. Diese ändert sich nur mit dem Programm »keybgr«.

Einer der wichtigsten Befehle ist »device«. Dieses Wort bedeutet soviel wie »Gerät« beziehungsweise »Vorrichtung«. Dieser Befehl dient dazu, zusätzliche Hardware zu installieren und zu optimieren. Dazu gehört die Maus sowie Laufwerke und Festplatten.

Mit der Zeile

```
device = ansi.sys
```

laden Sie den Treiber »ansi.sys«. Die Befehlseingabe entnehmen Sie Tabelle 1. Sie können damit aber auch andere Geräte oder Treiber installieren. So richten Sie mit der Eingabe

```
device = vdisk.sys 64
```

eine RAM-Diskette mit 64 KByte Speicherkapazität ein. Der Treiber »vdisk.sys«, der bei verschiedenen Herstellern »ramdrive.sys« heißt, muß im Hauptverzeichnis stehen, oder aber Sie teilen dem Betriebssystem extra mit, wo es ihn findet. Sollte der Treiber im Unterverzeichnis »system« Ihrer Festplatte C stehen, so geben Sie ein

```
device = c:\sys\vdisk.sys 64
```

Benötigen Sie eine größere RAM-Disk, wählen Sie einfach einen höheren Wert. Vergessen Sie dabei aber nicht, daß diese »Pseudo-Diskette« wertvollen Speicherplatz verbraucht und immer 64 KByte kleiner sein muß als der Hauptspeicher.

Sie können aber nicht nur die Größe, sondern auch die Anzahl der Sektoren und der Directory-Einträge angeben. Die dazugehörige Syntax entnehmen Sie bitte Tabelle 2. Die Größe eines Sektors muß entweder 128, 256, 512 oder 1024 betragen, andernfalls wird die von Ihnen angegebene Zahl auf- oder abgerundet. Mit dieser Zahl legen Sie fest, wie groß oder klein die Sektoren sind, in die MS-DOS eine Datei zerlegt, die in der RAM-Disk abgelegt werden soll. Bei kleineren Zahlen verbrauchen Sie weniger Speicherplatz pro Datei, allerdings dauern dann die Zugriffe etwas länger.

Bei größeren Werten verhält es sich genau umgekehrt. Sollten Sie sehr viele Dateien in der RAM-Disk ablegen, empfiehlt es sich, auch die Anzahl der Einträge im Directory zu vergrößern. Ohne eine Eingabe Ihrerseits nimmt MS-DOS immer den Wert 64 an, maximal sind 1024 Einträge im Hauptverzeichnis erlaubt. Die Option »/e« ist nur für AT-Besitzer wichtig, die Ihre RAM-Disk in einem Speicherbereich über 1 MByte einrichten wollen. Übrigens, Sie können auch mehrere RAM-Disks gleichzeitig definieren – sofern Sie dazu genügend Hauptspeicher zur Verfügung.

Das Betriebssystem vergibt die Laufwerksbezeichnungen selbständig. Trägt zum Beispiel das letzte Laufwerk die Bezeichnung B, so erhält die erste virtuelle Diskette automatisch die Bezeichnung C und die nächste den Buchstaben C zugeteilt. Nun aber vom Pseudolaufwerk zur echten Hardware. Mit dem Befehl »drivparm« lassen sich die standardmäßigen Gerätetreiber umdefinieren. Dieser Befehl wird nur selten gebraucht, da die meisten Laufwerke oder Festplatten ohne spezielle Anpassung angeschlossen werden können. Die Daten für die einzelnen Parameter (Tabelle 3) entnehmen Sie bitte der technischen Beschreibung des Geräteherstellers, oder Sie bitten Ihren Fachhändler um Auskunft.

Normalerweise akzeptiert MS-DOS bis zu sechs angeschlossene Massenspeicher (Festplatte, Streamer, RAM-Disk, Diskettenlaufwerke etc.) und vergibt auch dementsprechend die Laufwerksbezeichnungen. Benötigen Sie nun aber mehr »Buchstaben«, behelfen Sie sich mit dem Befehl:

```
lastdrv = f
```

Sie können dadurch bis zum Buchstaben »Z« Laufwerksnamen vergeben. Interessant ist dieser Befehl hauptsächlich für Benutzer, die mit dem MS-DOS Befehl »subst« Daten in ein bestimmtes Laufwerk »umlenken«.

Wenn Sie des öfteren mit Datenbanken arbeiten, sind Sie sicherlich schon mit dem Befehl »Files« in Berührung gekommen. Dieser Befehl legt die maximale Anzahl gleichzeitig geöffneter Dateien fest. Ohne eine Eingabe ist der Wert auf 8 festgelegt. Davon benötigt MS-DOS selbst den

```
device = (Laufwerk)(Pfad)(Treiber)(Parameter)
Beispiel
device =                    c:\system\ansi.sys
```

**Tabelle 1. Die Angaben für einen Device-Treiber in der Datei »config.sys«**

```
device = (Laufwerk)(Pfad)vdisk.sys(Größe)(Sektoren)(Directory)(/e)
Beispiel
device = c:\system\vdisk.sys 64 128 64
```

**Tabelle 2. Eine RAM-Diskette mit dem Device-Befehl**

```
driveparm = (Laufwerk)(/f:)(/h:)(/s:)(/t:)(/c)(/n)

(Laufwerk)  Geben Sie hier eine Zahl ein, so entspricht zum Beispiel 4 Laufwerk d
(/f)        Art des Geräts, eine Zahl von 0 bis 7
0           320 oder 360 KByte
1           1,2 MByte
2           720 KByte
3           8-Zoll-Laufwerk, einfache Dichte
4           8-Zoll-Laufwerk, doppelte Dichte
5           Festplatte
6           Bandgerät
7           1,44 MByte
(/h:)       Wieviele Schreib/Leseköpfe das Gerät hat
(/s:)       Anzahl der Sektoren einer Spur
(/t:)       Spuren pro Seite
(/c)        Es muß ein Hebel umgelegt werden, damit das Gerät richtig arbeiten kann
(/n)        Das Speichermedium kann nicht entfernt werden, zum Beispiel eine Festplatte
```

**Tabelle 3. Alle Angaben für den Befehl driveparm«**

Zugriff auf fünf Dateien, die restlichen drei sind für Anwendungsprogramme vorgesehen. Jede Datei, die zu den acht dazukommt, benötigt 48 Byte Speicherplatz. Für Datenbankanwendungen inklusive einiger speicherresidenter Utilities sollten Sie

```
files = 30
```

in Ihre »config.sys« eintragen. Der Maximalwert liegt bei 255, Werte unter 8 werden nicht berücksichtigt.

Sie sehen, daß Sie mit der Datei »config.sys« Ihre »Periperiegeräte« optimal an Ihren Computer anpassen können.

Die Mühe, diese Datei sorgfältig aufzubauen, lohnt sich also für ein problemloses Arbeiten.

(dm)

# Programmieren wie von Zauberhand

**Der direkte Zugriff auf den Tastaturpuffer ist für bestimmte Probleme die eleganteste Lösung. Unsere Listings in Basic und Pascal zeigen, wie es gemacht wird und unterstützen Sie beim Programmieren des Tastaturpuffers.**

Die Tastatur ist das wichtigste Eingabemedium des PC, denn ohne Tastatur wäre es kaum möglich, mit dem Computer zu arbeiten. Oder könnten Sie sich eine Bedienung Ihres PC ausschließlich über eine Maus vorstellen?

Die Entwickler des BIOS (Basic Input Output System) haben eine raffinierte Möglichkeit gefunden, die Eingaben des Benutzers so effektiv wie möglich zu verwalten. Sie richteten einen sogenannten Tastaturpuffer ein, in dem jedes eingegebene Zeichen gespeichert wird.

Der Tastaturpuffer befindet sich im Arbeitsspeicher des PC und besitzt eine Speicherkapazität von 15 Zeichen, damit ein Programm auch dann mehrere Tastendrücke bearbeiten kann, wenn es während der Eingabe dieser Zeichen andere Aufgaben erledigt. Der Tastaturpuffer arbeitet nach dem FIFO-Prinzip (»first in, first out«), was bedeutet, daß die zuerst gedrückte Taste auch zuerst an das Betriebssystem weitergegeben wird.

Der Puffer liegt im Arbeitsspeicher im Bereich von 41E bis 43Dhex (2x16 Byte). Das heißt aber nicht, daß die gedrückten Tasten mit der ersten Speicherstelle des Puffers beginnend in chronologischer Reihenfolge abgelegt werden. Es existieren vielmehr zwei 16-Bit-Zeiger, die auf den logischen Anfang und das logische Ende des Tastaturpuffers zeigen. Der Zeiger für den Pufferanfang ist in Speicherzelle 41A und 41Bhex und der Zeiger für das Pufferende in Speicherzelle 41C und 41Dhex abgespeichert. Um die korrekte Speicheradresse zu ermitteln, muß zum Inhalt der beiden Zeiger der Wert 400hex für das Segment addiert werden. Die Funktion des Tastaturpuffers läßt sich am besten anhand eines praktischen Beispiels demonstrieren. Nehmen Sie einmal an, daß der Tastaturpuffer leer ist und beide Pufferzeiger auf die Adresse 41Ehex zeigen. Nun drücken Sie die Taste [1]. Dadurch legt das BIOS den Wert 49 (der ASCII-Wert der Ziffer 1) in der Speicherstelle 41Ehex und den zugehörigen Scancode in 41Fhex ab. Auch wird der Zeiger, der das Ende des Puffers angibt, um zwei erhöht. Der Scancode gibt die Position einer Taste auf der Tastatur wieder und kann je nach verwendetem Computertyp differerieren. Allerdings fragen nur wenige Programme diese spezielle Information ab.

## Der Tastaturpuffer des PC kann bis zu 15 Tastendrücke aufnehmen und verarbeiten

Wenn der Tastaturpuffer mit 15 Zeichen vollgeschrieben ist (zwei Byte am Ende des Puffers werden nicht benutzt), gibt es zwei Möglichkeiten. Die erste Möglichkeit besteht darin, daß eine weitere Taste gedrückt wird, wenn der Puffer noch vollgeschrieben ist. In diesem Fall protestiert das BIOS, indem es einen Piepston ausgibt und dadurch einen Überlauf des Tastaturpuffers anzeigt. Die Annahme des eingegebenen Zeichens wird verweigert.

Ist dagegen beim nächsten Tastendruck wieder ein Eintrag im Tastaturpuffer freigeworden, so wird der Zeiger, der die Position des ersten Zeichens angibt, korrigiert. Das BIOS nutzt die beiden freien Byte, um das neu eingegebene Zeichen abzuspeichern, und paßt daraufhin auch den Zeiger für das Pufferende an die neue Situation an. Aus diesem Grund kann durchaus der Fall auftreten, daß der Wert des Zeigers, der auf das logisch letzte Zeichen weist, kleiner ist, als der Wert des Zeigers, der auf den logischen Pufferanfang zeigt.

Die dargelegten Grundlagen über den Tastaturpuffer befähigen Sie nun, direkt auf diesen Puffer zuzugreifen. Deshalb stellen wir Ihnen im folgenden je ein Basic- und Pascal-Programm vor, das den Tastaturpuffer nach Ihren Wünschen programmiert, und ein zweites Basic-Programm demonstriert ein Anwendungsbeispiel.

Listing 1 veranschaulicht, wie sich der Tastaturpuffer des PC in Basic programmieren läßt, und Listing 2 zeigt ein Programmierbeispiel in Turbo Pascal. Den Scancode berücksichtigen diese beiden Programme allerdings nicht, weil er von anderer Software in der Regel nicht ausgewertet wird. Ist dies doch der Fall, so funktionieren die beiden Programme nicht korrekt. Verwenden Sie beispielsweise das in eine Com-Datei umgewandelte Pascal-Programm, um bei der Textverarbeitung Word zur Voreinstellung in der MS-DOS-Kommandoebene ein Escape-Zeichen in den Tastaturpuffer zu schreiben, so wird lediglich das Zeichen mit dem ASCII-Wert 27 im Textfenster angezeigt, weil Word zu den wenigen Programm gehört, die neben dem ASCII-Wert auch den Scancode einer Taste auswerten.

Einige Anwendungsprogramme wandeln auch den Code der Esc-Taste in das ASCII-Zeichen 0 um. Aus diesem Grund müssen Sie ausprobieren, in welcher Situation sich die aufgeführten Programme verwenden lassen.

Das Basic-Programm (Listing 1) enthält einige Programmzeilen, die einer genaueren Erläuterung bedürfen. Der Programmteil von Zeile 1050 bis Zeile 1130 nimmt eine Zeichenkette bis zu einer Länge von 15 Zeichen entgegen. Dabei werden alle Zeichen akzeptiert, die einen ASCII-Wert besit-

```
[BVO] 1000 '
[OTG] 1010 'Direkte Eingabe von Zeichen in den Tastatur
      puffer
[HVO] 1020 '
[GH1] 1030 CLS
[AS2] 1040 DEF SEG=0
[D41] 1050 LOCATE 1,34:PRINT"Tastaturpuffer":LOCATE 2,3
      4:PRINT"==============":LOCATE 4,2:PRINT"Geben Si
      e bitte die Zeichen ein, die in den Tastaturpuffe
      r geschrieben werden":LOCATE 7,17:PRINT"sollen (B
      eenden durch Drücken der <ESC>-Taste):"
[TS7] 1060 LOCATE 15,33 : PRINT EINGABE$ : "# "
[0L8] 1070 A$ = INKEY$ : IF A$ = "" THEN GOTO 1070
[2EG] 1080 IF ASC(A$) = 13 THEN A$ = CHR$(190) 'Eingabe
       von <RETURN> umleiten
[FBD] 1090 IF ASC(A$) = 27 THEN GOTO 1170       'Beenden
       der Eingabe
[SCM] 1100 IF ASC(A$) =  8 THEN IF LEN(EINGABE$) > 0 TH
      EN EINGABE$ = LEFT$(EINGABE$,LEN(EINGABE$)-1) : G
      OTO 1060
[MBE] 1110 IF (ASC(A$) < 32) OR ((ASC(A$) > 127) AND (A
      SC(A$) <> 190)) THEN GOTO 1070
[SRC] 1120 IF LEN(EINGABE$)<15 THEN EINGABE$ = EINGABE$
       + A$ : GOTO 1060
[ML2] 1130 GOTO 1070
[TVO] 1140 '
[K7A] 1150 'Eingegebene Daten verarbeiten
[ZVO] 1160 '
[775] 1170 LAENGE = LEN(EINGABE$)
[NX2] 1180 DEF SEG = 0
[L97] 1190 POKE 1050,30 ' Pufferkopf
[5U8] 1200 POKE 1052,30+(LAENGE*2) ' Pufferende
[HD7] 1210 FOR ZAEHLER=1 TO LAENGE*2 STEP 2
[1A9] 1220 CODE = ASC(MID$(EINGABE$,(ZAEHLER+1)/2,1))
[G96] 1230 IF CODE = 190 THEN CODE = 13
[NKK] 1240 POKE 1053+ZAEHLER,CODE 'Zeichen in den Tasta
      turpuffer schreiben
[0L3] 1250 NEXT ZAEHLER
[RK1] 1260 END
```

**Listing 1. Auf diese Weise läßt sich der Tastaturpuffer in Basic programmieren**

```
[DK8]    0: program tastaturpuffer;
[000]    1:
[EMO]    2: var
[HC7]    3:   parameter : string[15];
[H36]    4:   zaehler   : integer;
[000]    5:
[X4D]    6: procedure pufferschreiben(var zeichen);
[000]    7:
[EMO]    8:   var
[ZR9]    9:     anfang  : integer    absolute 0:1050;
[5Z8]   10:     ende    : integer    absolute 0:1052;
[BV5]   11:     zaehler : integer;
[9CB]   12:     zeile   : string[15] absolute zeichen;
[000]   13:
[BH1]   14:   begin
```

```
[XU2]   15:     anfang:=30;
[AB7]   16:     ende  :=30+length(zeile)*2;
[XT9]   17:     for zaehler:=1 to length(zeile) do
[9ZC]   18:       mem[0:1054+(zaehler-1)*2]:=ord(zeile[
           zaehler]);
[TUO]   19:   end;
[000]   20:
[BH1]   21: begin
[M89]   22:   parameter:=chr(13)+'*.pas'+chr(13);
[349]   23:   pufferschreiben(parameter);
[5TO]   24: end.
```

**Listing 2. Dieses Programm demonstriert die Programmierung des Tastaturpuffers in Turbo Pascal**

zen, der höher als 31 ist. Zusätzlich sind Sie in der Lage, ein Wagenrücklauf-Zeichen (ASCII-Wert 13) durch Drücken der Enter-Taste einzugeben. Beendet wird die Eingabe durch das Betätigen der Esc-Taste.

Nachdem Sie die Zeichenkette angegeben haben, wird diese in dem Bereich von Zeile 1170 bis 1250 in den Tastaturpuffer geschrieben. Dabei speichert das Unterprogramm zuerst die Länge der Zeichenkette in den Zeigern ab, die den Anfang und das Ende des Tastaturpuffers spezifizieren. Die folgende Schleife ab Zeile 1210 legt die ASCII-Werte der einzelnen Zeichen in Zweierschritten nacheinander im Tastaturpuffer ab; der Scan-Code wird nicht beachet.

Das Turbo-Pascal-Programm (Listing 2) arbeitet nach dem gleichen Prinzip. Der einzige Unterschied besteht darin, daß die zu bearbeitende Zeichenkette mit Hilfe der Variablen »parameter« an das Unterprogramm beziehungsweise die Prozedur »pufferschreiben« übergeben wird.

Listing 3 zeigt ein interessantes Anwendungsbeispiel für die Programmierung des Tastaturpuffers. Indem die ersten Programmzeilen den periodischen Aufruf eines Unterprogramms bewirken, ist es möglich, Daten in den Tastaturpuffer zu schreiben, die länger als 15 Zeichen sind. Voraussetzung dafür ist allerdings, daß sich die zu bearbeitende Zeichenkette in der Variablen »puffer$« befindet. Erreicht wird dieses Ziel durch die Aktivierung des Timer-Interrupts in Zeile 2, der jede Sekunde die Prozedur ab Zeile 60030 aufruft. Innerhalb dieser Routine schalten die Befehle von Zeile 60030 bis Zeile 60070 die Tastatur so lange ab, bis die Variable »puffer$« vollständig in den Tastaturpuffer geschrieben worden ist.

Das wird dadurch erreicht, daß das Unterprogramm jeweils die ersten 15 Zeichen der Variablen in den Puffer schreibt und die Zeichenkette entsprechend um 15 Zeichen kürzt.

Zwischen dem Aufruf des Unterprogramms und dem eigentlichen Unterprogramm können Sie nun ein eigenes Programm plazieren, das sich die Programmierung des Tastaturpuffers zunutze macht. Mit dem vorgestellten Programm sind Sie beispielsweise in der Lage, eine Makro-Sprache zu entwickeln, in der jeder Befehl den Tastaturpuffer mit der gewünschten Zeichenfolge füllt. Oder Sie können eine Hilfsfunktion programmieren, die in einem Programm automatisch Abfragen mit Standardwerten beantworten, wenn der Anwender nicht mehr weiter weiß. Sie sehen, das Spektrum für Ihre Anwendungen läßt sich durch die Programmierung des Tastaturpuffers beträchtlich erweitern.

(Markus Zietlow/ma)

```
[1P4] 1 ON TIMER(1) GOSUB 60030
[U62] 2 TIMER ON
[BVO] 1000 '
[LCN] 1010 'In dem Bereich zwischen Zeile 3 und 59999 k
      ann das Hauptprogramm stehen
[HVO] 1020 '
[751] 60000 '
[L6H] 60010 'Unterprogramm, das den Tastaturpuffer prog
      rammiert
[F51] 60020 '
[KFA] 60030 OUT 97,204 'Tastatur ausschalten
[R25] 60040 LAENGE = LEN(PUFFER$)
[RO3] 60050 DEF SEG = 0
[KZ6] 60060 IF LAENGE = 0 THEN GOTO 60170
[5X8] 60070 IF PEEK(1050) <> PEEK(1052) THEN GOTO 60180
[157] 60080 IF LAENGE>15 THEN LAENGE = 15
```

```
[AL7] 60090 POKE 1050,30 ' Pufferkopf
[PT8] 60100 POKE 1052,30+(LAENGE*2) ' Pufferende
[ZP7] 60110 FOR ZAEHLER=1 TO LAENGE*2 STEP 2
[X29] 60120 CODE = ASC(MID$(PUFFER$,(ZAEHLER+1)/2,1))
[CGK] 60130 POKE 1053+ZAEHLER,CODE 'Zeichen in den Tast
      aturpuffer schreiben
[RK3] 60140 NEXT ZAEHLER
[9DF] 60150 IF LEN(PUFFER$) > 15 THEN PUFFER$ = MID$(PU
      FFER$,16) ELSE PUFFER$ = ""
[AV2] 60160 GOTO 60180
[G6C] 60170 OUT 97,76'Tastatur wieder einschalten
[392] 60180 RETURN
```

**Listing 3. Ein Anwendungsbeispiel für die Programmierung des Tastaturpuffers**


**DIE VIER* Utilities vermissen Sie sicher schon lange...**

**RAMfrei**

entfernt **selektiv** residente Programme (per Tastendruck) und liefert freien Speicher (auch EMS) für hungrige Software.

**Nottaste**

Rechner abgestürzt? Programm aufgehängt? Ein Tastendruck, und Sie können sicher weiterarbeiten! Ein Muß für Programmierer und RAM-Disk-User!

**WS_PATH**

erweitert Wordstars Horizont. Bearbeiten Sie Ihre Texte ab jetzt in **allen** angeschlossenen Laufwerken und Verzeichnissen!

**Treiber**

Universeller Druckertreiber. Einfache und flexible Zeichenkonvertierung für parallele und serielle Druckerausgabe.

DIE VIER ZUSAMMEN inkl. deutsche Anleitung DM **99,–**

*VIER neue Utilities für DOS 2.xx und 3.xx für PC/XT/AT-Kompatible

Preis incl. Versandkosten, 2 Tage Lieferzeit

**Andreas Vehrenberg**
**Krefelder Str. 290**
**4156 Willich 1**

**02154/ 4624**

# Tastatureingaben gespart

**Immer wenn Sie anfangen, mit Turbo Pascal zu arbeiten, sind es immer wieder dieselben Tasten, die Sie betätigen müssen: Fehlermeldungen laden – die Arbeitsdatei anwählen. Mit dem Assemblerprogramm »push.com« fällt dies weg.**

Sicher haben Sie es auch schon bei anderen oder sich selbst beobachtet. Der Computer wird gestartet, mit der »autoexec.bat« laden Sie das gewünschte Programm, und dann folgen die unvermeidlichen und sich immer wiederholenden Routineeingaben über die Tastatur. Mit dem Programm »push.com« (Listing) können Sie nun diese Eingaben an den Tastaturpuffer übergeben und sozusagen Eingabeanforderungen selbsttätig ausführen lassen. Als Parameter können Sie eine Zeichenkette bis zu 15 Zeichen angeben. Nach dem Aufruf von »Push« behandelt das Programm diese Zeichenkette nämlich so, als würden Sie sie selbst von Hand in die Tastatur eingegeben.

Nachdem Sie den Quelltext eingetippt haben, dürfen Sie nicht vergessen, nach dem Linklauf die Datei mit »exe2bin« in eine Com-Datei umzuwandeln. Das fertige Programm belegt nur 165 Byte, ist also kein »Speicherfresser«.

Diese Zeichenkette kann alle ASCII-Zeichen enthalten. Control-Zeichen müssen Sie mit einem Control-Präfix (also mit »↑«) kennzeichnen. So ist beispielsweise der Ausdruck »↑M« gleichbedeutend mit der Taste [Return]. Das ASCII-Zeichen »↑« wiederum können Sie durch »↑↑« darstellen.

Die Anwendung des Programms ist vor allem im Zusammenspiel mit Batch-Dateien sinnvoll. Sie übergeben mit Hilfe von »push.com« dem Programm die nötigen Eingaben, die Sie normalerweise über die Tastatur eingeben. Folgende Befehle können Sie jeweils durch eine Batch-Datei ausführen lassen:

```
push yrdatei↑m
turbo
```

bewirkt, daß Sie Turbo Pascal aufrufen, die Datei mit den Fehlermeldungen, also »turbo.msg« laden sowie den Programmquellcode »datei.pas« laden, compilieren und zur Ausführung bringen. Es lassen sich natürlich noch andere Routinearbeiten mit Hilfe von »push.com« automatisieren. Ein weiteres Beispiel für eine solche Batch-Datei:

```
push h
chess
```

Mit diesen Zeilen erreichen Sie, daß beim Starten des Spiels »Psion Chess« automatisch die zum Spielen notwendige Hercules-Grafikkarte ausgewählt wird.

```
push 3
td
```

bewirkt dasselbe bei dem Programm »Test Drive«.
So wird »push.com« auf einfache Weise Bestandteil Ihr Batchdateien-Systems, und Sie ersparen sich dadurch unnötige Tipparbeiten. Oder Sie schreiben sich neue Stapeldateien zur Automatisierung Ihrer Programmierarbeiten, die routinemäßig immer wiederkehren.

(Peter Mandrella/gh)

```
[U59]    0: title PUSH.COM Batch Enhancement
[T52]    1: page  65,120
[000]    2:
[902]    3: org     100h
[000]    4:
[0H4]    5: code    segment
[W45]    6: assume  cs:code
[000]    7:
[2G4]    8: start   proc far
[7T2]    9:         mov  ax,40h
[VM2]   10:         mov  ds,ax
[8C4]   11:         mov  cl,es:[80h]
[592]   12:         cmp  cl,2
[XB2]   13:         jb   error
[NT2]   14:         mov  bx,82h
[X52]   15:         dec  cl
[792]   16:         mov  ch,0
[A66]   17: lp1:    mov  dl,es:[bx]
[292]   18:         inc  bx
[HK2]   19:         cmp  dl,'^'
[DD2]   20:         jnz  send
[X52]   21:         dec  cl
[MA2]   22:         or   cl,cl
[352]   23:         jz   ende
[3Y4]   24:         mov  dl,es:[bx]
[NF2]   25:         cmp  dl,'0'
[9C2]   26:         jz   pfeil
[Z43]   27:         and  dl,0dfh
[VN2]   28:         sub  dl,40h
[9C4]   29: pfeil:  inc  bx
[NU4]   30: send:   call append
[FM2]   31:         loop lp1
[G35]   32: ende:   mov  ax,4c00h
[4K1]   33:         int  21h
[NV4]   34: error:  mov  ax,cs
[CP2]   35:         add  ax,10h
[VM2]   36:         mov  ds,ax
[8F5]   37:         mov  dx,offset text
[492]   38:         mov  ah,9
[4K1]   39:         int  21h
[DY4]   40:         jmp  short ende
[R23]   41: start   endp
[000]   42:
[GD3]   43: append  proc
```

```
[7M2]   44:         push bx
[4R4]   45:         mov  bx,ds:[1ch]
[7M2]   46:         push bx
[KV2]   47:         cmp  bx,3ch
[ME2]   48:         jb   write
[GW2]   49:         mov  bx,1ch
[XD4]   50: write:  inc  bx
[292]   51:         inc  bx
[SN4]   52:         cmp  bx,ds:[1ah]
[172]   53:         jz   full
[RK4]   54:         mov  ds:[1ch],bx
[KA2]   55:         pop  bx
[DV4]   56:         mov  ds:[bx],dl
[KA2]   57:         pop  bx
[SM0]   58:         ret
[S24]   59: full:   pop  bx
[1M2]   60:         push ax
[DM2]   61:         push cx
[FE2]   62:         in   al,61h
[843]   63:         and  al,0fch
[AM2]   64:         mov  cx,300
[PL0]   65:         cli
[CU3]   66: flp1:   xor  al,2
[252]   67:         out  61h,al
[DM2]   68:         push cx
[2G2]   69:         mov  cx,72
[T44]   70: flp2:   loop flp2
[SA2]   71:         pop  cx
[0M2]   72:         loop flp1
[CU2]   73:         mov  cx,5000
[Y44]   74: flp3:   loop flp3
[PM0]   75:         sti
[SA2]   76:         pop  cx
[CA2]   77:         pop  ax
[KA2]   78:         pop  bx
[SM0]   79:         ret
[GD3]   80: append  endp
[000]   81:
[TZC]   82: text:   db 0dh,0ah,'parameter missing',0dh,
            0ah,'$'
[000]   83:
[UL2]   84: code    ends
[5P2]   85:         end  start
```

**Dieses Assemblerlisting erspart Ihnen Tastatureingaben**

# Der MS-DOS-Interrupt 21hex (Teil 1)

## Die CP/M-kompatiblen Funktionen 00hex bis 37hex

**Der MS-DOS-Interrupt 21hex ist einer der wichtigsten Interrupts, weil sich über ihn fast alle Funktionen aufrufen lassen, die das Betriebssystem MS-DOS zur Verfügung stellt. Der Interrupt 21hex ist damit für den Programmierer ein wichtiges Hilfsmittel, um schnelle und effektive Programme zu schreiben.**

Die Funktionen des MS-DOS-Interrupt 21hex unterteilen sich in CP/M- und Xenix-kompatible Funktionen. Die CP/M-kompatiblen Funktionen lassen sich über die Funktionsnummern 00hex bis 37hex aufrufen, und die Xenix-kompatiblen Funktionen folgen ab der Nummer 38hex. Die Funktionsnummer wird beim Aufruf einer Funktion jeweils im AH-Register übergeben. Dieser erste Beitrag über den MS-DOS-Interrupt 21hex beschäftigt sich ausschließlich mit den CP/M-kompatiblen Funktionen, die Xenix-kompatiblen folgen im zweiten Teil.

Im folgenden sind die CP/M-kompatiblen Funktionen des MS-DOS-Interrupt 21hex in der Reihenfolge ihrer Numerierung aufgeführt und werden mit Angabe der Ein- und Ausgabeparameter erläutert.

### Funktion 00hex: Programm beenden

Die Funktion 00hex des Interrupt 21hex (»Terminate Program«) dient dazu, den Ablauf des aktuellen Programms zu unterbrechen und in das aufrufende Programm zurückzukehren. Das ist in vielen Fällen die Kommandoebene des Betriebssystems MS-DOS, die alle zur Verfügung stehenden Dateien verwaltet.

Bevor die Funktion 00hex die Kontrolle an das aufrufende Programm übergibt, restauriert sie die drei Interruptvektoren, deren Inhalt vor dem Start des Programms im Programm Segment Präfix (PSP) gespeichert wurde. Der PC stellt den Arbeitsspeicher, den das beendete Programm belegt, nach dem Funktionsaufruf anderen Programmen zur Verfügung und schreibt den Inhalt aller Dateipuffer auf den angesprochenen Datenträger.

Soweit es vermeidbar ist, sollten Sie anstatt der Funktion 00hex die Funktion 4Chex des Interrupt 21hex benutzen, da Exe-Programme die Funktion 00hex nicht aufrufen können.

Beim Aufruf der Funktion 00hex muß das AH-Register notwendigerweise den Wert 0 enthalten. Im CS-Register legen Sie die Segmentadresse des zugehörigenPSP ab.

### Funktion 01hex: Zeichen lesen und ausgeben

Die Funktion 01hex liest ein Zeichen vom Standard-Eingabegerät (Voreinstellung: Tastatur), gibt es auf dem

Standard-Ausgabegerät (Voreinstellung: Bildschirm) aus und übergibt es zusätzlich im AL-Register an das aufrufende Programm. Wenn Sie [Ctrl-C] drücken, führt MS-DOS den Interrupt 23hex aus.

Steht zum Zeitpunkt des Funktionsaufrufs kein Zeichen am Standard-Eingabegerät zur Verfügung, so wartet die Funktion auf eine Eingabe. Das Standard-Eingabegerät ist nicht notwendigerweise die Tastatur und das Standard-Ausgabegerät nicht unbedingt der Bildschirm. Da Sie die Ein- und Ausgabekanäle umleiten können, ist es durchaus möglich, beispielsweise die serielle Schnittstelle als Ein- und einen Drucker als Ausgabegerät zu definieren.

Wenn Sie die Funktion 01hex aufrufen, muß das AH-Register den Wert 01hex enthalten. Nach der Rückkehr aus der Funktion enthält das AL-Register den ASCII-Wert des eingelesenen Zeichens. Der Inhalt der Register AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES und des Kennzeichenregisters ändert sich durch den Funktionsaufruf nicht.

**Funktion 02hex: Zeichen auf das Standard-Gerät ausgeben**

Die Funktion 02hex gibt ein Zeichen, das das aufrufende Programm im DL-Register übergibt, auf das Standard-Ausgabegerät aus. Wenn Sie [Ctrl-C] drücken, wird der Interrupt 23hex ausgeführt.

Ist das Standard-Ausgabegerät der Bildschirm, dann behandelt das Betriebssystem alle ASCII-Zeichen, deren Code kleiner als 32 ist, als Steuerzeichen. Ansonsten gibt die Funktion 02hex die Zeichen als ASCII-Werte aus. Das Standard-Ausgabegerät ist nicht unbedingt der Bildschirm. Weil Sie den Ausgabekanal umleiten können, ist es durchaus möglich, zum Beispiel einen Drucker als Ausgabegerät zu definieren.

Rufen Sie die Funktion 02hex auf, so muß das AH-Register den Wert 02hex und das DL-Register den ASCII-Wert des auszugebenden Zeichens enthalten. Nach der Rückkehr aus der Funktion enthalten alle Register einschließlich des Kennzeichenregisters ihre ursprünglichen Werte.

**Funktion 03hex: Zeichen von serieller Schnittstelle einlesen**

Mit der Funktion 03hex lesen Sie ein Zeichen von der seriellen Schnittstelle ein. Die Funktion 03hex übergibt dieses Zeichen im AL-Register an das aufrufende Programm. Wenn Sie [Ctrl-C] drücken, wird der Interrupt 23hex ausgeführt.

Steht zum Zeitpunkt des Funktionsaufrufs kein Zeichen an der seriellen Schnittstelle zur Verfügung, so wartet die Funktion auf eine Eingabe.

Die serielle Schnittstelle ist nicht unbedingt die Geräteeinheit mit der Bezeichnung »com1«. Wenn Sie die Einheit mit dem MS-DOS-Befehl »mode« umleiten, empfängt die Funktion 03hex die Daten über eine andere Schnittstelle. Haben Sie die Parameter der Schnittstelle nicht eingestellt, setzt MS-DOS ein Stopbit, keine Parität und 2400 Baud Übertragungsrate bei einer Datenlänge von acht Bit pro Byte als Standardwert fest.

Beim Aufruf der Funktion 03hex muß das AH-Register den Wert 03hex enthalten. Nach der Rückkehr aus der Funktion enthält das AL-Register das empfangene Zeichen. Der Inhalt der Register AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES und des Kennzeichenregisters ändert sich nicht.

**Funktion 04hex: Zeichen auf die serielle Schnittstelle ausgeben**

Die Funktion 04hex ist das Gegenstück zur Funktion 03hex, denn sie gibt ein Zeichen, dessen ASCII-Wert im DL-Register stehen muß, an die serielle Schnittstelle aus. Die Funktion 04hex wartet solange, bis die Schnittstelle signalisiert, daß das Zeichen erfolgreich abgesandt wurde.

Die serielle Schnittstelle ist nicht unbedingt das Gerät mit der Bezeichnung »com1«. Wenn Sie die Ausgabe mit dem Mode-Befehl umleiten, sendet die Funktion 04hex die Daten an eine andere Schnittstelle. Haben Sie die Parameter der Schnittstelle nicht eingestellt, so sind die voreingestellten Standardwerte ein Stopbit, keine Parität und 2400 Baud Übertragungsrate bei einer Datenlänge von acht Bit pro Byte.

Die Tastenkombination [Ctrl-C] bewirkt den Aufruf des Interrupt 23hex.

Rufen Sie die Funktion 04hex auf, dann muß das AH-Register den Wert 04hex und das DL-Register den ASCII-Wert des auszugebenden Zeichens enthalten. Nach der Rückkehr aus der Funktion enthalten alle Register einschließlich des Kennzeichenregisters ihre ursprünglichen Werte.

Es ist ratsam, an Stelle der Funktionen 03hex und 04hex die entsprechenden BIOS-Routinen des Interrupt 14hex zu nutzen, weil diese den Status der angewählten Schnittstelle berücksichtigen und ihn bei Bedarf an das aufrufende Programm übermitteln.

**Funktion 05hex: Zeichen an die parallele Schnittstelle übergeben**

Die Funktion 05hex gibt ein Zeichen, dessen ASCII-Wert das aufrufende Programm in das DL-Register schreiben muß, an die parallele Schnittstelle aus. Die Funktion wartet so lange, bis die parallele Schnittstelle signalisiert, daß sie das Zeichen übergeben hat.

Die Schnittstelle ist nicht von vornherein das Gerät mit der Bezeichnung »prn« beziehungsweise »lpt1«. Leiten Sie die Ausgabe mit dem Mode-Befehl um, so übergibt die Funktion 05hex die Daten an die eingestellte Ausgabeeinheit. Falls Sie die Tastenkombination [Ctrl-C] betätigen, wird der Interrupt 23hex aufgerufen.

Sie wählen die Funktion 05hex aus, indem Sie in das AH-Register den Wert 05hex und in das DL-Register den ASCII-Wert des auszugebenden Zeichens schreiben.

Nach der Rückkehr aus der Funktion sind alle Register einschließlich des Kennzeichenregisters unverändert.

Beachten Sie bitte, daß die Funktionen 03hex, 04hex und 05hex keinen Fehlercode und keinen Statuswert an das aufrufende Programm übergeben.

**Funktion 06hex: Zeichenein- und -ausgabe**

Die Funktion 06hex stellt zwei Unterfunktionen zur Verfügung, die entsprechend dem Inhalt des DL-Registers aufgerufen werden. Enthält DL einen Wert ungleich FFhex, so gibt die Funktion den Inhalt auf das Standard-Ausgabegerät aus. Andernfalls liest die Funktion 06hex ein Zeichen vom Standard-Eingabegerät ein und übergibt dieses im AL-Register an das aufrufende Programm. Wenn zum Zeitpunkt des Funktionsaufrufs kein Zeichen am Standard-Eingabegerät zur Verfügung steht, setzt die Funktion das Zero-Flag des Kennzeichenregisters, ansonsten löscht sie es.

Als Standard-Eingabegerät benutzt die Funktion 06hex nicht notwendigerweise die Tastatur, oder als Standard-Ausgabegerät den Bildschirm. Weil Sie den Ein- und Ausgabekanal umleiten können, sind Sie in der Lage, andere Einheiten als Ein- und Ausgabegeräte zu definieren.

Wenn Sie die Funktion 06hex aufrufen, muß das AH-Register den Wert 06hex enthalten. Nach der Rückkehr aus der Funktion enthält beim lesenden Zugriff das AL-Register den ASCII-Wert des eingelesenen Zeichens, wenn das Zero-Flag gelöscht ist. Der Inhalt der Register AH, BX, CX, DX, SI, DI, BP, CS, DS, SS und ES ändert sich durch den Funktionsaufruf nicht.

**Funktion 07hex: Warten auf Zeicheneingabe**

Die Funktion 07hex liest ein Zeichen vom Standard-Eingabegerät ein und übergibt es im AL-Register an das

aufrufende Programm. Wenn zum Zeitpunkt des Funktionsaufrufs kein Zeichen am Standard-Eingabegerät zur Verfügung steht, wartet die Funktion auf eine Eingabe. Das Standard-Eingabegerät ist nicht unbedingt die Tastatur, weil Sie den Eingabekanal umleiten können.

Wenn Sie die Funktion 07hex aufrufen, muß das AH-Register den Wert 07hex enthalten. Nach der Rückkehr aus der Funktion enthält das AL-Register den ASCII-Wert des eingelesenen Zeichens. Der Inhalt der Register AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES und des Kennzeichenregisters ändert sich durch den Funktionsaufruf nicht.

**Funktion 08hex: Warten auf Zeicheneingabe unter Berücksichtigung von [Ctrl-C]**

Die Funktion 08hex liest ebenfalls ein Zeichen vom Standard-Eingabegerät ein und übergibt dieses im AL-Register an die aufrufende Routine. Steht zum Zeitpunkt des Funktionsaufrufs kein Zeichen am Standard-Eingabegerät zur Verfügung, so wartet die Funktion auf eine Eingabe. Das Standard-Eingabegerät ist nicht unbedingt die Tastatur, weil Sie den Eingabekanal umleiten können.

Der einzige Unterschied der Funktion 08hex zur Funktion 07hex besteht darin, daß die Funktion 08hex den Interrupt 23hex ausführt, wenn die Tastenkombination [Ctrl-C] gedrückt wird.

Beim Aufruf der Funktion 08hex muß das AH-Register den Wert 08hex enthalten. Nachdem die Funktion ausgeführt ist, enthält das AL-Register den ASCII-Wert des eingelesenen Zeichens. Der Inhalt der Register AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES und des Kennzeichenregisters ändert sich nicht.

Alle bisher vorgestellten Routinen arbeiten mit einzelnen Zeichen, deren ASCII-Wert die Funktion jeweils in einem Register übernimmt. Das ist zwar in vielen Fällen sinnvoll und oft auch notwendig, andererseits erschwert es den Umgang mit Daten erheblich, wenn MS-DOS mehrere Zeichen hintereinander ausgeben oder übernehmen soll. Die folgenden Funktionen sind dafür vorgesehen, Ein- und Ausgaben von längeren Zeichenketten durchzuführen.

**Funktion 09hex: Zeichenkette ausgeben**

Die Funktion 09hex gibt eine Zeichenkette, deren Start durch die Register DS und DX angegeben wird, auf das Standard-Ausgabegerät aus. Ist das Standard-Ausgabegerät der Bildschirm, so bearbeitet die Funktion 09hex alle ASCII-Zeichen, deren Code kleiner als 32 ist, als Steuerzeichen.

Das Standard-Ausgabegerät ist nicht notwendigerweise der Bildschirm. Weil Sie den Ausgabekanal umleiten können, ist es durchaus möglich, beispielsweise einen Drucker oder Plotter als Ausgabegerät zu definieren.

Das Ende der Zeichenkette markieren Sie mit dem Dollar-Zeichen (»$«), das die Funktion 09hex nicht mehr anzeigt.

Um eine Zeichenkette auszugeben, muß das AH-Register den Wert 09hex enthalten. Die Position der Zeichenkette übergeben Sie über den Zeiger DS:DX an die Funktion 09hex. Nach der Ausführung der Routine enthalten alle Register einschließlich des Kennzeichenregisters ihre ursprünglichen Werte.

**Funktion 0Ahex: Zeichenkette einlesen**

Die Funktion 0Ahex liest eine Zeichenkette vom Standard-Eingabegerät ein und speichert sie in einem zuvor definierten Eingabepuffer ab. Die Funktion wird erst dann beendet, wenn der Benutzer die Eingabe der Zeichen durch Drücken der Enter-Taste bestätigt.

Empfängt die Funktion 0Ahex Daten, obwohl der Eingabepuffer voll ist, so erzeugt Sie einen Warnton in Form eines kurzen Piepsers.

Wenn die Tastatur das Standard-Eingabegerät ist, dann sind Sie in der Lage, die eingegebene Zeichenkette mit den dafür vorgesehenen Sondertasten zu editieren. Die Tastatur ist jedoch nicht unbedingt das Standard-Eingabegerät, weil Sie den Eingabekanal umleiten können. Wird [Ctrl-C] gedrückt, führt MS-DOS den Interrupt 23hex aus.

Die Position des Tastaturpuffers übergeben Sie mit dem Zeiger DS:DX an die Funktion. Um eine Zeichenkette einzulesen, muß das AH-Register den Wert 0Ahex enthalten.

Nach Bearbeitung der Funktion 0Ahex enthält der Eingabepuffer ab dem dritten Byte die eingegebene Zeichenkette. Das zweite Byte gibt die Länge der Zeichenkette an, ohne das abschließende Enter-Zeichen zu berücksichtigen. Vor dem Aufruf der Prozedur geben Sie im ersten Byte des Tastaturpuffers an, wieviele Bytes die Funktion einlesen und ab der dritten Position des Tastaturpuffers abspeichern soll. Die Längenangabe berücksichtigt das abschließende Betätigen der Enter-Taste.

Die Funktion 0Ahex verändert keinen Registerwert.

**Funktion 0Bhex: Eingabeeinheit auf bereitstehendes Zeichen überprüfen**

Mit der Funktion 0Bhex stellen Sie fest, ob ein Zeichen am Standard-Eingabegerät bereitsteht. Das Ergebnis dieser Überprüfung erhalten Sie im AL-Register. Signalisiert das Standard-Eingabegerät zum Zeitpunkt des Funktionsaufrufs, daß kein Zeichen anliegt, so speichert die Funktion 0Bhex im Register AL den Wert 0 ab, andernfalls den Wert FFhex.

Das Standard-Eingabegerät ist nicht unbedingt die Tastatur, weil Sie den Eingabekanal umleiten können.

Wenn die Tastenkombination [Ctrl-C] betätigt wird, ruft MS-DOS den Interrupt 23hex auf.

Beim Aufruf der Funktion 0Bhex muß das AH-Register den Wert 0Bhex enthalten. Nachdem die Funktion ausgeführt ist, enthält das AL-Register den Status des Standard-Eingabegerätes. Der Inhalt der Register AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES und des Kennzeichenregisters ändert sich durch den Funktionsaufruf nicht.

**Funktion 0Chex: Eingabepuffer löschen und Zeicheneingabe**

Die Funktion 0Chex löscht den Standard-Eingabepuffer und ruft eine der Funktionen zur Zeichen-Eingabe auf (01hex, 06hex, 07hex, 08hex und 0Ahex). Die aufgerufene Funktion liest daraufhin ein Zeichen vom Standard-Eingabegerät und übergibt es im AL-Register an das aufrufende Programm.

Das Standard-Eingabegerät ist nicht unbedingt die Tastatur, weil Sie den Eingabekanal umleiten können.

Wenn Sie für die zweite aufzurufende Funktion einen falschen Parameter übergeben, kehrt die Funktion 0Chex in das aufrufende Programm zurück, ohne eine Aktion durchzuführen.

Das Löschen des Eingabepuffers lohnt sich natürlich nur dann, wenn die Tastatur das Standard-Eingabegerät ist. Andere Anwendungsgebiete sind wenig sinnvoll.

Wenn Sie die Funktion 0Chex aufrufen, muß das AH-Register den Wert 0Chex und das AL-Register die Nummer der aufzurufenden Funktion enthalten. Nach der Rückkehr aus der Funktion enthält das AL-Register den ASCII-Wert des eingelesenen Zeichens.

Die Ein- und Ausgabeparameter entnehmen Sie bitte den entsprechenden Funktionen (01hex, 06hex, 07hex, 08hex, 0Ahex). Der Inhalt der Register AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES und des Kennzeichenregisters ändert sich durch den Funktionsaufruf nicht.

**Funktion 0Dhex: Dateipuffer leerschreiben**

Die Funktion 0Dhex schreibt alle in einem MS-DOS-internen Dateipuffer gespeicherten Daten auf das gewünschte Ausgabegerät. In jedem Fall gibt die Funktion die Zeichen als ASCII-Werte aus. Achten Sie darauf, daß die Funktion keine Datei schließt und keinen Datenkanal freigibt. Es ist notwendig, daß Sie diesen Vorgang selbst durchführen.

Um einen Dateipuffer leerzuschreiben, muß das AH-Register den Wert 0Dhex enthalten. Die Funktion löst die gestellte Aufgabe ohne die Angabe weiterer Parameter. Nach der Rückkehr aus der Funktion enthalten alle Register einschließlich des Kennzeichenregisters ihre ursprünglichen Werte.

**Funktion 0Ehex: Laufwerk auswählen**

Mit der Funktion 0Ehex wählen Sie das aktuelle Laufwerk aus. Alle weiteren Dateizugriffe, bei denen keine Laufwerksangabe spezifiziert ist, erfolgen von nun an über das in dieser Funktion angegebene Standardlaufwerk.

Die gewünschte Laufwerksnummer schreiben Sie in das AL-Register. Dabei steht der Wert 0 für das Laufwerk A, der Wert 1 für das Laufwerk B, der Wert 2 für die Festplatte C und so weiter.

Nach dem Aufruf schreibt die Funktion 0Ehex alle geöffneten Dateien auf den Datenträger zurück.

Um ein Laufwerk auszuwählen, müssen Sie in das AH-Register den Wert 0Ehex schreiben. Nachdem die Funktion ausgeführt ist, sind mit Ausnahme von AL alle Register einschließlich des Kennzeichenregisters unverändert.

Das AL-Register enthält die Anzahl der verfügbaren logischen Laufwerke. Doch berücksichtigen Sie bei weiteren Befehlen diese Angabe bitte nicht, weil sie in einigen Fällen zu ungenau ist. Der Wert im AL-Register berücksichtigt nämlich auch RAM-Disks und die verschiedenen Partitionen der Festplatte.

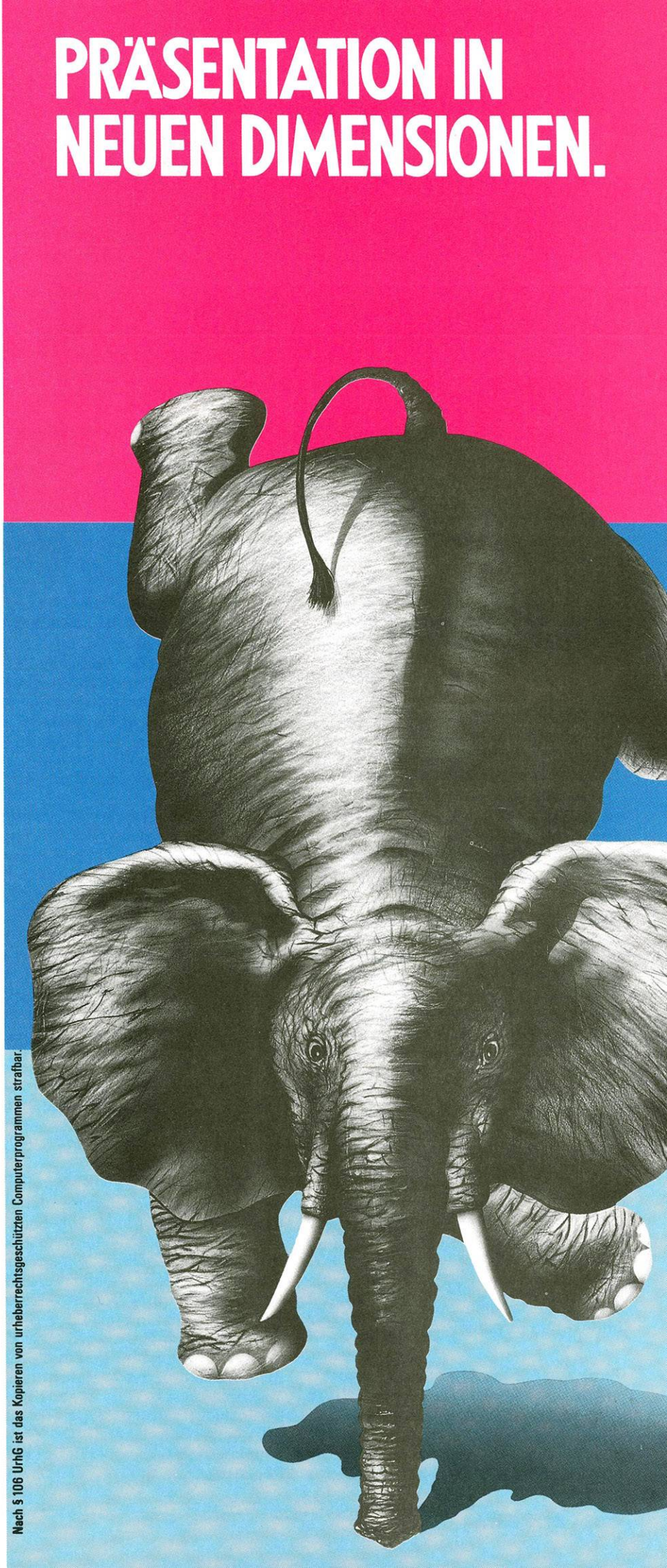
Bei den nun folgenden Dateifunktionen können Sie die Wildcards »?« und »*« benutzen, um einen Dateinamen anzugeben.

**Funktion 0Fhex: Datei öffnen**

Mit der Funktion 0Fhex versuchen Sie, eine Datei zu öffnen. Als Ergebnis erhalten Sie im AL-Register einen Wert, der Ihnen Aufschluß darüber gibt, ob die Aktion erfolgreich abgeschlossen wurde. Findet die Funktion 0Fhex zum Zeitpunkt des Auf-



PRÄSENTATION IN NEUEN DIMENSIONEN.



Nach § 106 UrhG ist das Kopieren von urheberrechtsgeschützten Computerprogrammen strafbar.

# MICROSOFT CHART 3.0.

Präsentationen sollen weder der Auslöser für Kopfstände sein, noch sollen sie Kopfzerbrechen verursachen.
Um sie so einfach wie möglich zu gestalten, präsentieren wir Ihnen MICROSOFT CHART 3.0. Ein komplettes Präsentationsgrafik-Paket: Text, Zahlen und die Umsetzung von Zahlenmaterial in optisch ansprechende Diagramme.
In der neuen Version mit noch mehr Möglichkeiten und noch mehr Bedienungskomfort. Durch ein im Programm enthaltenes Lernprogramm einfach zu erlernen – mit der MICROSOFT MAUS extrem bequem und schnell zu bedienen.
Alle Befehle können mit der Maus ausgewählt und durchgeführt werden – selbst bei der weiteren Bearbeitung im Diagramm. Änderungen sind darum direkt und ohne Umwege möglich, was für Sie eine sensationelle Zeitersparnis bedeutet und Sie maushoch überlegen macht.
Lernen Sie MICROSOFT CHART 3.0 kennen. Denn es stellt alles auf den Kopf, was Sie bisher gekannt haben.

- **Dreidimensionale Darstellungen.**
- **Statistische Auswertung von Daten.**
- **Standardeinstellung für Textformate.**
- **Automatische Positionierung von Texten.**
- **Freie Linien.**
- **Glättung von Liniendiagrammen.**
- **Bis zu 16 Diagramme gleichzeitig darstellbar (Fenstertechnik).**
- **Überlagern und kombinieren von Diagrammen.**
- **Direkte Datenübernahme von MICROSOFT MULTIPLAN, WKS, ASCII, SYLK, DIF, dBase II/III.**

MS/DOS CBT 320/KB 3½ 5¼

Microsoft®
ZUKUNFT DER SOFTWARE

COUPON

**Bitte senden Sie mir Informationsmaterial zu MICROSOFT CHART 3.0. Ich nutze Software:** ☐ privat ☐ beruflich/Branche _____
**Mein Rechner:** ☐ MS DOS ☐ MS OS/2 ☐ Macintosh
**Bitte senden Sie den Coupon an:** Microsoft GmbH · Erdinger Landstraße 2 · 8011 Aschheim-Dornach
**Absender nicht vergessen.**

DOS E 4/88


rufs im Inhaltsverzeichnis einen Eintrag zu dem angegebenen Dateinamen, dann wird im AL-Register der Wert 0 gespeichert und die Datei geöffnet.
Die Funktion 0Fhex setzt darauf die Datensatzlänge auf 128 Byte und trägt im Dateisteuerblock (FCB) das Zeit- und Datumsattribut der ursprünglichen Datei ein. Bei sequentiellen Dateien müssen Sie die sequentielle Datensatzlänge (Offset 20hex) und bei relativen Dateien die relative Datensatzlänge (Offset 21hex) setzen.
Wenn im Inhaltsverzeichnis kein Eintrag für die gesuchte Datei vorhanden ist oder die Datei das Dateiattribut »versteckt« beziehungsweise »System« trägt, dann wird der Inhalt des AL-Registers auf FFhex gesetzt.
Um eine Datei zu öffnen, muß das AH-Register den Wert 0Fhex enthalten und die Registerkombination DS:DX auf einen unbenutzten FCB zeigen. Nachdem die Funktion ausgeführt ist, informiert das AL-Register darüber, ob die Datei im Inhaltsverzeichnis eingetragen ist oder nicht. Der Inhalt der Register AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES und des Kennzeichenregisters ändert sich durch den Funktionsaufruf nicht.

**Funktion 10hex: Datei schließen**
Mit der Funktion 0Fhex schließen Sie eine Datei. Als Ergebnis erhalten Sie im AL-Register einen Wert, der Ihnen Aufschluß darüber gibt, ob die Aktion erfolgreich durchgeführt wurde. Findet die Funktion 10hex zum Zeitpunkt des Aufrufs den Namen der gesuchten Datei im Inhaltverzeichnis, so wird das AL-Register auf 0 gesetzt und die Datei geschlossen.
Die Funktion 10hex vergleicht auch das Zeit- und Datumsattribut im spezifizierten FCB mit der ursprünglichen Datei und paßt dieses bei Bedarf an. Wenn der angegebene Dateieintrag im Inhaltsverzeichnis nicht vorhanden oder mit dem Dateiattribut »versteckt« beziehungsweise »System« gekennzeichnet ist, wird der Inhalt des AL-Registers auf FFhex gesetzt, um einen Fehler anzuzeigen.
Um eine Datei zu schließen, muß das AH-Register den Wert 10hex enthalten und die Registerkombination DS:DX auf den gewünschten FCB zeigen. Nachdem das Betriebssystem die Funktion ausgeführt hat, informiert das AL-Register darüber, ob die Datei im Inhaltsverzeichnis eingetragen ist oder nicht. Der Inhalt der Register AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES und des Kennzeichenregisters ändert sich durch den Funktionsaufruf nicht.

**Funktion 11hex: Ersten Eintrag im Inhaltsverzeichnis suchen**

Die Funktion 11hex sucht den ersten Eintrag im Inhaltsverzeichnis des aktuellen Verzeichnisses. Als Ergebnis erhalten Sie im AL-Register einen Wert, der Sie informiert, ob die Aktion erfolgreich beendet wurde. Findet die Funktion 11hex zum Zeitpunkt des Aufrufs einen Eintrag im Inhaltverzeichnis, dann speichert sie im AL-Register den Wert 0 und legt im FCB der ungeöffneten Datei den Dateinamen ab.

Handelt es sich dabei um einen normalen FCB, so enthält das erste Byte die Laufwerksangabe und die weiteren 32 Byte den Eintrag des Inhaltsverzeichnisses. Ist der Dateisteuerblock dagegen ein erweiterter FCB, so speichert die Funktion 11hex zusätzlich vor dem eigentlichen Eintrag eine Zeichenfolge ab, die aus dem Wert FFhex und fünf Null-Byte besteht und die das Attributbyte des Dateieintrags abschließt.

Wenn dagegen der gesuchte Eintrag nicht im Inhaltsverzeichnis vorhanden ist, wird das AL-Register auf den Wert FFhex gesetzt.

Beim Aufruf der Funktion 11hex muß das AH-Register den Wert 11hex enthalten und die Registerkombination DS:DX auf den ungeöffneten FCB zeigen. Nachdem die Funktion ausgeführt ist, informiert das AL-Register darüber, ob die Datei im Inhaltsverzeichnis eingetragen ist oder nicht. Der Inhalt der Register AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES und des Kennzeichenregisters ändert sich durch den Funktionsaufruf nicht.

**Funktion 12hex: Nächsten Eintrag im Inhaltsverzeichnis suchen**

Die Funktion 12hex sucht die nächste Datei im Inhaltsverzeichnis des aktuellen Verzeichnisses, nachdem Sie mit Funktion 11hex den ersten Eintrag gefunden haben. Als Ergebnis erhalten Sie im AL-Register einen Wert, der Sie informiert, ob die Suche von Erfolg gekrönt ist.

Findet die Funktion 12hex zum Zeitpunkt des Aufrufs einen Eintrag im Inhaltverzeichnis, dann wird im Register AL der Wert 0 gespeichert und im FCB der ungeöffneten Datei der Dateinamen abgelegt.

Handelt es sich um einen normalen FCB, so enthält das erste Byte die Laufwerksangabe und die weiteren 32 Byte den Eintrag im Inhaltsverzeichnis. Ist der Dateisteuerblock hingegen ein erweiterter FCB, dann speichert das Betriebssystem zusätzlich vor dem eigentlichen Eintrag eine Zeichenfolge, die aus dem Wert FFhex und fünf Null-Bytes besteht und die das Attributbyte des Dateieintrags abschließt.

Wenn die gesuchte Datei nicht gefunden wurde oder kein weiterer Eintrag vorhanden ist, setzt die Funktion 12hex den Inhalt des AL-Registers auf den Wert FFhex.

Beim Aufruf der Funktion 12hex muß das AH-Register den Wert 12hex enthalten und die Registerkombination DS:DX auf den ungeöffneten FCB zeigen. Nachdem die Funktion ausgeführt ist, informiert das AL-Register darüber, ob die Datei im Inhaltsverzeichnis eingetragen ist oder nicht. Der Inhalt der Register AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES und des Kennzeichenregisters ändert sich durch den Funktionsaufruf nicht.

**Funktion 13hex: Datei löschen**

Mit der Funktion 13hex löschen Sie eine oder mehrere Dateien. Als Ergebnis erhalten Sie im AL-Register einen Wert, der Aufschluß darüber gibt, ob die Funktion die Dateien gelöscht hat.

Findet die Funktion 13hex zum Zeitpunkt des Aufrufs den Namen der Datei im Inhaltsverzeichnis, dann wird die Datei gelöscht und im AL-Register der Wert 0 abgelegt. Sie können mit der Funktion 13hex allerdings keine Verzeichnisse löschen.

Wenn die zu löschende Datei dagegen nicht im aktuellen Verzeichnis vorhanden ist, erhält das AL-Register den Wert FFhex zugewiesen.

Um eine Datei zu löschen, muß das AH-Register den Wert 13hex enthalten und die Registerkombination DS:DX auf den benötigten FCB zeigen. Nachdem die Funktion ausgeführt ist, informiert das AL-Register darüber, ob die Datei im Inhaltsverzeichnis eingetragen ist oder nicht. Der Inhalt der Register AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES und des Kennzeichenregisters ändert sich durch den Funktionsaufruf nicht.

**Funktion 14hex: Nächsten Datensatz einlesen**

Die Funktion 14hex dient dazu, den folgenden Datensatz in den Diskettenübertragungsbereich (DTA) einzulesen. Nach dem Aufruf der Funktion 14hex gibt das AL-Register an, ob die Aktion ausgeführt werden konnte. Wurde nach dem Aufruf der Funktion 14hex ein Datensatz ohne Komplikationen eingelesen, so erhält das AL-Register den Wert 0 zugewiesen. Hat die Funktion dagegen das Dateiende erreicht, so enthält das AL-Register den Wert 1. Ist AL auf 2 gesetzt, dann liegt der Diskettenübertragungsbereich (DTA) am Ende eines Segments, und der eingelesene Datensatz geht über das Segmentende hinaus. Beinhaltet AL den Wert 3, so befindet sich am Dateiende nur der Teil eines Datensatzes, nicht aber ein kompletter Datensatz. In diesem Fall wird der DTA mit Null-Bytes auf die vorgegebene Länge aufgefüllt.

Die Funktion 14hex setzt den Dateizeiger nach dem Einlesen der Informationen automatisch auf den nächsten Datensatz. Beim Aufruf dieser Funktion muß das AH-Register den Wert 14hex enthalten und die Registerkombination DS:DX auf den gewünschten FCB zeigen. Nachdem das Betriebssystem die Funktion ausgeführt hat, enthält das AL-Register den Fehlercode. Der Inhalt der Register AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES und des Kennzeichenregisters ändert sich nicht.

**Funktion 15hex: Nächsten Datensatz schreiben**

Die Funktion 15hex schreibt den nächsten im DTA enthaltenen Datensatz in eine Datei. Nach dem Aufruf dieser Funktion gibt das AL-Register an, ob die Aktion mit Erfolg durchgeführt wurde. Hat die Funktion 15hex einen Datensatz ohne Schwierigkeiten in die Datei geschrieben, dann enthält AL den Wert 0.

Wenn der Datenträger vollgeschrieben ist, bekommt das AL-Register den Wert 1 zugewiesen. Ist AL auf 2 gesetzt, liegt der DTA am Ende eines Segments, und der geschriebene Datensatz geht über das Segmentende hinaus.

Wenn Sie die Funktion 15hex auswählen, muß das AH-Register den Wert 15hex enthalten und die Registerkombination DS:DX auf den FCB zeigen. Nachdem die Funktion ausgeführt ist, enthält das AL-Register den Fehlercode. Der Inhalt der Register AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES und des Kennzeichenregisters ändert sich nicht.

**Funktion 16hex: Anlegen einer Datei**

Mit der Funktion 16hex versuchen Sie, eine neue Datei anzulegen. Als Ergebnis erhalten Sie im AL-Register einen Wert, der Ihnen Aufschluß darüber gibt, ob diese Aktion erfolgreich abgeschlossen wurde. Findet die Funktion 16hex zum Zeitpunkt des Aufrufs einen Eintrag im Inhaltsverzeichnis, wird die Länge der bestehenden Datei auf 0 gesetzt. Existiert die Datei noch nicht, so erzeugt die Funktion eine neue Datei. In beiden Fällen wird im AL-Register der Wert 0 gespeichert und die Datei geöffnet. Konnte die Funktion 16hex die gewünschte Datei dagegen nicht anlegen, beispielsweise weil das Inhaltsverzeichnis oder der Datenträger bereits gefüllt ist, enthält das AL-Register den Wert FFhex.

Um eine neue Datei anzulegen, muß das AH-Register den Wert 16hex enthalten und die Registerkombination DS:DX auf einen FCB zeigen. Nachdem die Funktion ausgeführt ist, informiert das AL-Register darüber, ob die Datei im Inhaltsverzeichnis erzeugt wurde oder nicht. Der Inhalt der Register AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES und des Kennzeichenregisters ändert sich durch den Funktionsaufruf nicht.

**Funktion 17hex: Datei umbenennen**

Mit der Funktion 17hex benennen Sie eine oder mehrere Dateien um. Als Ergebnis erhalten Sie im AL-Register einen Wert, der angibt, ob dieser Vorgang erfolgreich abgeschlossen wurde. Findet die Funktion 17hex zum Zeitpunkt des Aufrufs den Namen der umzubenennenden Datei im Inhaltverzeichnis, so legt sie im AL-Register den Wert 0 ab und übernimmt den neuen Namen für die Datei. Die Funktion vergleicht das Zeit- und Datumsattribut im spezifizierten FCB mit der ursprünglichen Datei und paßt es bei Bedarf an. Der FCB enthält neben dem ursprünglichen Dateinamen ab der 16. Stelle die Gerätebezeichnung und den neuen Namen der Datei.

Wenn Sie für die Funktion 17hex Wildcards benutzen, ändert die Funktion die Namen einer ganzen Dateigruppe. Wenn die Funktion 17hex die spezifizierte Datei nicht gefunden hat oder der neue Name bereits existiert, wird im AL-Register der Wert FFhex gespeichert.

Um eine Datei umzubenennen, muß das AH-Register den Wert 17hex enthalten und die Registerkombination DS:DX auf den FCB zeigen. Nachdem die Funktion ausgeführt ist, informiert das AL-Register darüber, ob die Datei den neuen Namen erhalten hat oder nicht. Der Inhalt der Register AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES und des Kennzeichenregisters ändert sich durch den Funktionsaufruf nicht.

**Funktion 19hex: Gerätebezeichnung des aktuellen Laufwerks lesen**

Mit der Funktion 19hex ermitteln Sie die Gerätebezeichnung des aktuellen Laufwerks. Die Funktion 19hex übergibt die Laufwerksnummer im Register AL. Dabei steht der Wert 0 für Laufwerk A, der Wert 1 für Laufwerk B, der Wert 2 für Festplatte C, und so weiter.

Sie wählen die Funktion 19hex aus, indem Sie in das AH-Register den Wert 19hex schreiben. Nach der Rückkehr aus der Funktion sind mit Ausnahme von AL alle Register einschließlich des Kennzeichenregisters unverändert. Das AL-Register enthält die Gerätenummer des aktuellen Laufwerks.

**Funktion 1Ahex: Neue Adresse des Diskettenübertragungsbereichs (DTA) bestimmen**

Die Funktion 1Ahex legt eine neue Adresse für den DTA fest. Diese Funktion wird nur dann korrekt bearbeitet, wenn die Adreßeingabe im erlaubten Bereich liegt. Die neue Adresse des DTA übergeben Sie mit dem Zeiger DS:DX an die Funktion.

Unter normalen Umständen beginnt der DTA ab Adresse 80hex des Programm Segment Präfix (PSP). Eine Verlegung ist sinnvoll, wenn der DTA eine Länge erhalten soll, die 128 Byte überschreitet. Mit der Funktion 2Fhex ermitteln Sie übrigens die aktuelle Adresse des DTA.

Um die Funktion 1Ahex aufzurufen, muß das AH-Register den Wert 1Ahex enthalten. Die Funktion verändert keinen Registerwert.

**Funktion 1Bhex: Daten des aktuellen Laufwerks ermitteln**

Mit der Funktion 1Bhex ermitteln Sie Daten über das aktuelle Laufwerk. Die Funktion übergibt die Anzahl der Sektoren pro Cluster im AL-Register und die Gesamtanzahl der Cluster im DX-Register. Die Registerkombination DS:BX zeigt auf den Media-Descriptor, der den physikalischen Aufbau des Datenträgers bestimmt. Dabei steht der Wert F8hex für eine Festplatte, F9hex für eine zweiseitige Diskette mit 15 Sektoren pro Spur, FChex für eine einseitige Diskette mit neun Sektoren pro Spur, FDhex für eine zweiseitige Diskette mit neun Sektoren pro Spur, FEhex für eine einseitige Diskette mit acht Sektoren pro Spur und schließlich FFhex für eine zweiseitige Diskette mit acht Sektoren pro Spur.

Sie wählen die Funktion 1Bhex aus, indem Sie in das AH-Register den Wert 1Bhex schreiben. Der Inhalt der Register AH, CX, SI, DI, BP, CS, SS, ES und des Kennzeichenregisters ändert sich durch den Funktionsaufruf nicht.

**Funktion 1Chex: Daten eines bestimmten Laufwerks ermitteln**

Die Funktion 1Chex ermittelt Daten über ein bestimmtes Laufwerk. Vor dem Aufruf der Funktion muß das DL-Register die Laufwerksnummer des zu überprüfenden Registers enthalten. Dabei steht der Wert 0 für das Standardlaufwerk, der Wert 1 für das Laufwerk A, der Wert 2 für das Laufwerk B, und so weiter.

Die Funktion 1Chex gibt die Anzahl der Sektoren pro Cluster im AL-Register, die Gesamtanzahl der Cluster im DX-Register und die Anzahl der Bytes pro Cluster im CX-Register zurück. Die Registerkombination DS:BX zeigt auf den Media-Descriptor, der den physikalischen Aufbau des Datenträgers bestimmt. Dabei steht der Wert F8hex für eine Festplatte, F9hex für eine zweiseitige Diskette mit 15 Sektoren pro Spur, FChex für eine einseitige Diskette mit neun Sektoren pro Spur, FDhex für eine zweiseitige Diskette mit neun Sektoren pro Spur, FEhex für eine einseitige Diskette mit acht Sektoren pro Spur und schließlich FFhex für eine zweiseitige Diskette mit acht Sektoren pro Spur.

Wenn die angegebene Laufwerksnummer ungültig ist, enthält das AL-Register nach dem Beenden der Funktion den Wert FFhex.

Sie wählen die Funktion 1Bhex aus, indem Sie in das AH-Register den Wert 1Bhex schreiben. Der Inhalt der Register AH, SI, DI, BP, CS, SS, ES und des Kennzeichenregisters ändert sich durch den Funktionsaufruf nicht.

Die Funktion 1Dhex bis 20hex sind für MS-DOS reserviert.

**Funktion 21hex: Wahlfreien Datensatz lesen**

Die Funktion 21hex dient dazu, den Datensatz einer relativen Datei in den DTA einzulesen. Nach dem Aufruf der Funktion gibt das AL-Register an, ob die Aktion mit Erfolg durchgeführt werden konnte. Hat die Funktion 21hex nach dem Aufruf einen Datensatz ohne Komplikationen eingelesen, enthält das AL-Register den Wert 0.

Wenn die Funktion 21hex beim Lesen der Datensätze das Dateiende erreicht hat, enthält das AL-Register den Wert 1. Ist AL gleich zwei, so liegt der DTA am Ende eines Segments, und der eingelesene Datensatz geht über das Segmentende hinaus. Wenn in AL der Wert 3 abgelegt ist, befindet sich am Dateiende nur der Teil eines Datensatzes, nicht aber ein kompletter Datensatz. In diesem Fall wird der Datenpuffer mit Null-Bytes auf die vorgegebene Länge aufgefüllt. Vor dem Aufruf der Funktion 21hex speichern Sie im FCB die Position des einzulesenden Datenblocks ab. Beim Aufruf der Funktion muß das AH-Register den Wert 21hex enthalten und die Registerkombination DS:DX auf den gewünschten FCB zeigen. Nachdem die Funktion ausgeführt ist, enthält das AL-Register den Fehlercode. Der Inhalt der Register AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES und des Kennzeichenregisters ändert sich durch den Funktionsaufruf nicht.

**Funktion 22hex: Wahlfreien Datensatz schreiben**

Die Funktion 22hex schreibt den Datensatz einer relativen Datei in den DTA. Nach dem Aufruf der Funktion gibt das AL-Register an, ob die Aktion mit Erfolg durchgeführt wurde. Hat die Funktion einen Datensatz ohne Schwierigkeiten in die Datei geschrieben, dann enthält AL den Wert 0.

Wenn der Datenträger vollgeschrieben ist, beinhaltet das AL-Register den Wert 1. Ist AL auf 2 gesetzt, dann liegt der DTA am Ende eines Segments, und der geschriebene Datensatz geht über das Segmentende hinaus. Vor dem Aufruf der Funktion legen Sie im FCB die Position des zu schreibenden Datenblocks ab.

Wenn Sie die Funktion 22hex auswählen, muß das AH-Register den Wert 22hex enthalten und die Registerkombination DS:DX auf den FCB zeigen. Nachdem die Funktion ausgeführt ist, enthält das AL-Register den Fehlercode. Der Inhalt der Register AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES und des Kennzeichenregisters ändert sich durch den Funktionsaufruf nicht.

**Funktion 23hex: Größe einer Datei ermitteln**

Mit der Funktion 23hex ermitteln Sie die Größe einer Datei. Als Ergebnis erhalten Sie im AL-Register einen Wert, der angibt, ob der Vorgang erfolgreich abgeschlossen wurde. Findet die Funktion 23hex zum Zeitpunkt des Aufrufs die Datei im aktuellen Verzeichnis, so legt sie im AL-Register den Wert 0 ab und speichert die Anzahl der Datensätze ab der Adresse 21hex im FCB. Vor dem Aufruf der Funktion müssen Sie die Datensatzgröße in den FCB eintragen.

Wenn die Funktion 23hex die zu überprüfende Datei nicht gefunden hat, wird dem AL-Register der Wert FFhex zugewiesen.

Um die Größe einer Datei zu ermitteln, muß das AH-Register den Wert 23hex enthalten und die Registerkombination DS:DX auf den FCB zeigen. Nachdem die Funktion ausgeführt ist, informiert das AL-Register darüber, ob die Datei gefunden wurde oder nicht. Der Inhalt der Register AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES und des Kennzeichenregisters ändert sich durch den Funktionsaufruf nicht.

**Funktion 24hex: Position für den wahlfreien Datensatz bestimmen**

Die Funktion 24hex dient dazu, die Datensatznummer im FCB für den relativen Zugriff auf eine Datei auf den aktuellen Dateizeiger zu setzen.

Beim Aufruf dieser Funktion muß das AH-Register den Wert 24hex enthalten und die Registerkombination DS:DX auf den gewünschten FCB zeigen. Nachdem die Funktion ausgeführt ist, enthält das Register AL den Fehlercode. Der Inhalt der Register AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES und des Kennzeichenregisters ändert sich nicht.

**Funktion 25hex: Interruptvektor setzen**

Die Funktion 25hex setzt einen Interruptvektor auf eine neue Adresse, die den Startpunkt einer Interrupt-Serviceroutine markiert. Um Probleme zu vermeiden, sollten Sie solche Änderungen niemals direkt in den Speicher schreiben. Bevor Sie die Adresse eines Vektors ändern, ist es sinnvoll, die ursprüngliche Interruptadresse mit der Funktion 35hex auszulesen und für eine spätere Verwendung zu speichern.

Um einen Interruptvektor neu zu setzen, muß das AH-Register den Wert 25hex enthalten und die Registerkombination DS:DX auf die neue Interruptadresse zeigen. Der Inhalt der Register AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES und des Kennzeichenregisters ändert sich durch den Funktionsaufruf nicht.

**Funktion 26hex: Neues PSP anlegen**

Die Funktion 26hex versetzt Sie in die Lage, ein neues Programm-Segment-Präfix (PSP) zu erzeugen. Die Funktion 26hex erreicht das gewünschte Ergebnis dadurch, daß es das PSP des ausgeführten Programms an die vorgegebene Adresse kopiert. Sie übergeben die Segmentadresse des neuen PSP im DX-Register. Als Offset setzt die Funktion 26hex automatisch den Wert 0 voraus.

Um ein neues PSP anzulegen, muß das AH-Register den Wert 26hex enthalten. Nach der Rückkehr aus der Funktion 26hex enthalten alle Register einschließlich des Kennzeichenregisters ihre ursprünglichen Werte.

**Funktion 27hex: Mehrere wahlfreie Datensätze lesen**

Die Funktion 27hex dient dazu, mehrere Datensätze einer relativen Datei in den DTA zu lesen. Nach dem Aufruf dieser Funktion gibt das AL-Register an, ob die Aktion erfolgreich durchgeführt wurde. Hat die Funktion 27hex die Datensätze ohne Komplikationen eingelesen, enthält das AL-Register den Wert 0. Wenn die Funktion das Dateiende erreicht hat, ist das AL-Register auf den Wert 1 gesetzt. Beträgt der Inhalt von AL 2, so liegt der DTA am Ende eines Segments, und die eingelesenen Datensätze gehen über das Segmentende hinaus. Ist in AL der Wert 3 abgelegt, befindet sich am Dateiende nur der Teil eines Datensatzes, nicht aber ein kompletter Datensatz. In diesem Fall wird der Datenpuffer mit Null-Bytes auf die vorgegebene Länge aufgefüllt.

Vor dem Aufruf der Funktion 27hex speichern Sie im FCB die Position des ersten einzulesenden Datenblocks ab.

Beim Aufruf der Funktion 27hex muß das AH-Register den Wert 27hex enthalten und die Registerkombination DS:DX auf den gewünschten FCB zeigen. Das CX-Register gibt die Anzahl der einzulesenden Datensätze an. Nachdem die Funktion ausgeführt ist, enthält das AL-Register den Fehlercode und das CX-Register die Anzahl der gelesenen Datensätze. Der Inhalt der Register AH, BX, DX, SI, DI, BP, CS, DS, SS, ES und des Kennzeichenregisters ändert sich durch den Funktionsaufruf nicht.

**Funktion 28hex: Mehrere wahlfreie Datensätze schreiben**

Die Funktion 28hex schreibt mehrere Datensätze aus dem DTA in eine relative Datei. Nach dem Aufruf der Funktion 28hex gibt das AL-Register an, ob die Aktion erfolgreich war. Wurden die Datensätze ohne Schwierigkeiten in die Datei geschrieben, enthält das AL-Register den Wert 0. Wenn der Datenträger vollgeschrieben ist, so beträgt der Inhalt des AL-Registers 1. Ist AL auf 2 gesetzt, liegt der DTA am Ende eines Segments, und die geschriebenen Datensätze gehen über das Segmentende hinaus. Vor dem Aufruf der Funktion legen Sie im FCB die Position des zu schreibenden Datenblocks ab.

Wenn Sie die Funktion 28hex auswählen, muß das AH-Register den Wert 28hex enthalten, das CX-Register die Anzahl der zu schreibenden Datensätze angeben und die Registerkombination DS:DX auf den FCB zeigen. Nachdem die Funktion ausgeführt ist, enthält das AL-Register den Fehlercode und das CX-Register die Anzahl der geschriebenen Datensätze. Der Inhalt der Register AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES und des Kennzeichenregisters ändert sich durch den Funktionsaufruf nicht.

**Funktion 29hex: Dateinamen in den FCB übertragen**

Die Funktion 29hex dient dazu, einen Dateinamen in den Dateisteuerblock (FCB) zu schreiben. Der Dateiname besteht aus dem Namen zuzüglich der Extension und kann jedes ASCII-Zeichen darstellen; er muß nur mit dem Wert 0 abgeschlossen sein.

Beim Aufruf der Funktion 29hex muß das AH-Register den Wert 29hex ent-

halten und die Registerkombination DS:SI auf den Dateinamen zeigen. Die Position ES:DI ist die Startadresse des FCB einer ungeöffneten Datei. Das AL-Register enthält einen Suchcode und kontrolliert die Übertragung des Dateinamens. Ist das Bit 0 auf 1 gesetzt, ignoriert die Funktion 29hex die im Dateinamen enthaltenen Trennzeichen, und ist Bit 0 auf 0 gesetzt, bricht sie beim Auftreten dieser Zeichen ab und kehrt in das aufrufende Programm zurück.
Das Bit 1 im AL-Register bestimmt, ob das aktuelle (Bit gesetzt) oder das Standardlaufwerk (Bit gelöscht) angesteuert wird. Bit 2 ersetzt den Dateinamen durch acht Leerzeichen, wenn er unzulässige Zeichen enthält und das Bit gelöscht ist. Ist das Bit dagegen gesetzt, ändert die Funktion 29hex den Dateinamen nicht.
Bit 3 ersetzt die Extension durch drei Leerzeichen, wenn sie unzulässige Zeichen enthält und das Bit gelöscht ist. Ist das Bit gesetzt, ändert die Funktion die Erweiterung nicht.
Nachdem die Funktion 29hex ausgeführt ist, enthält das AL-Register Informationen über die verwendeten Daten. Ist kein Teil des Dateinamens ein Wildcard-Zeichen, wird dem AL-Register der Wert 0 zugewiesen. Besteht ein Teil des Dateinamens aus Wildcards, dann beinhaltet AL den Wert 1. Ist AL auf FFhex gesetzt, so ist die angegebene Gerätebezeichnung ungültig.
Die Registerkombination DS:SI zeigt auf das erste Byte nach dem Dateinamen. Der Inhalt der Register AH, BX, CX, DX, BP, CS, SS und des Kennzeichenregisters ändert sich durch den Funktionsaufruf nicht.

**Funktion 2Ahex: Datum ermitteln**

Mit der Funktion 2Ahex stellen Sie das aktuelle Datum fest. Sie erhalten im AL-Register eine Zahl, die den Wochentag anzeigt. Dabei steht »0« für Sonntag, »1« für Montag, »2« für Dienstag und so weiter. Das CX-Register übergibt das aktuelle Jahr, DH den Monat und DL den Tag.
Die Funktion 2Ahex benutzt den Uhrtreiber, um das aktuelle Datum zu ermitteln.
Um das aktuelle Datum zu lesen, müssen Sie in das AH-Register den Wert 2Ahex schreiben. Der Inhalt der Register AH, BX, SI, DI, BP, CS, DS, SS, ES und des Kennzeichenregisters ändert sich durch den Funktionsaufruf nicht.

**Funktion 2Bhex: Datum setzen**

Die Funktion 2Bhex wird eingesetzt, um das aktuelle Datum auf einen vorgegebenen Wert zu setzen. Im CX-Register übergeben Sie das aktuelle Jahr, in DH den Monat und in DL den Tag. Die Funktion 2Bhex benutzt den Uhrtreiber, um das aktuelle Datum zu setzen. Sind die übergebenen Parameter sinnvoll, so löscht die Funktion das alte Datum, aktualisiert es und legt im AL-Register den Wert 0 ab.
Wenn die Parameter, die Sie an die Funktion 2Bhex übergeben, dagegen ungültig sind, wird im AL-Register der Wert FFhex zurückgegeben.
Um das Datum zu setzen, müssen Sie in das AH-Register den Wert 2Ahex schreiben. Der Inhalt der Register AH, BX, SI, DI, BP, CS, DS, SS, ES und des Kennzeichenregisters ändert sich durch den Funktionsaufruf nicht.

**Funktion 2Chex: Systemzeit ermitteln**

Mit der Funktion 2Chex läßt sich die Systemzeit lesen. Wenn Sie diese Funktion aufrufen, erhalten Sie im CH-Register die Stunde, im CL-Register die Minute sowie in DH die Sekunde und in DL die Hundertstelsekunde der aktuellen Zeit zurück.
Die Funktion 2Chex benutzt den Uhrtreiber, um die aktuelle Systemzeit zu ermitteln.
Um die Systemzeit zu lesen, müssen Sie in das AH-Register den Wert 2Chex schreiben. Der Inhalt der Register AX, BX, SI, DI, BP, CS, DS, SS, ES und des Kennzeichenregisters ändert sich durch den Funktionsaufruf nicht.

**Funktion 2Dhex: Systemzeit setzen**

Die Funktion 2Dhex ist das Gegenstück zur Funktion 2Chex, denn sie wird benutzt, um die Systemzeit auf einen vorgegebenen Wert zu setzen.
Im CH-Register übergeben Sie die aktuelle Stunde, im CL-Register die Minute, in DH die Sekunde und in DL die Hundertstelsekunde. Auch die Funktion 2Dhex benutzt den Uhrtreiber, um eine neue Systemzeit zu setzen.
Sind die übergebenen Parameter für die neue Systemzeit sinnvoll, so löscht die Funktion 2Dhex die alte Zeit, setzt sie auf die neuen Werte und legt im AL-Register den Wert 0 ab. Wenn die Parameter, die Sie an die Funktion 2Dhex übergeben, dagegen ungültig sind, wird das AL-Register auf den Wert FFhex gesetzt.
Sie setzen die Systemzeit, indem Sie in das AH-Register den Wert 2Dhex schreiben. Der Inhalt der Register AH, BX, CX, DX, SI, DI, BP, CS, DS, SS, ES und des Kennzeichenregisters ändert sich nicht.

**Funktion 2Ehex: Verify-Flag setzen/löschen**

Auf Wunsch überprüft MS-DOS alle Daten, die es auf einen Datenträger geschrieben hat, auf ihre Richtigkeit. Diese Vorgehensweise ist für die Datensicherheit zwar förderlich, benötigt aber zusätzliche Zeit. Benutzen Sie hochwertige Disketten, so können Sie getrost auf diese Überprüfung verzichten. In der MS-DOS-Kommandoebene schalten Sie die Überprüfung mit »verify on« ein und mit »verify off« aus. In Assemblersprache läßt sich der gleiche Effekt durch das Setzen und Löschen des Verify-Flag über die Funktion 2Ehex erzielen.

Um das Verify-Flag zu setzen, müssen Sie vor dem Aufruf der Funktion 2Ehex den Wert 1 in das AL-Register schreiben, und um das Verify-Flag zu löschen muß dementsprechend der Wert 0 im AL-Register stehen.

Um die Funktion 2Ehex aufzurufen, muß das AH-Register den Wert 2Ehex enthalten. Nach der Beendigung der Funktion 2Egex enthalten alle Register einschließlich des Kennzeichenregisters ihre ursprünglichen Werte.

**Funktion 2Fhex: Adresse des DTA ermitteln**

Die Funktion 2Fhex stellt die Adresse des Diskettenübertragungsbereichs (DTA) fest. Beachten Sie, daß die angegebenen Werte nichts über die Größe des DTA aussagen. Die Adresse des DTA wird von der Funktion 2Fhex über den Zeiger ES:BX zurückgegeben.

Unter normalen Umständen beginnt der DTA ab Adresse 80hex des Programm Segment Präfix (PSP). Eine Abfrage der DTA-Position ist dann sinnvoll, wenn der Anwender den Übertragungsbereich verlegt hat, um für den DTA eine Länge zu erhalten, die größer als 128 Byte ist. Mit der Funktion 1Ahex legen Sie die Adresse des DTA übrigens neu fest.

Um die DTA-Adresse zu ermitteln, muß das AH-Register den Wert 2Fhex enthalten. Der Inhalt der Register AX, CX, DX, SI, DI, BP, CS, DS, SS und des Kennzeichenregisters ändert sich durch den Funktionsaufruf nicht.

**Funktion 30hex: MS-DOS-Versionsnummer lesen**

Mit der Funktion 30hex können Sie die Nummer des MS-DOS-Version abfragen. Nach dem Aufruf der Funktion 30hex wird im AL-Register die über- und im AH-Register die untergeordnete Versionsnummer übergeben. Bei der MS-DOS-Version 3.31 ist beispielsweise die Zahl 3 im AL-Register und der Wert 31 im AH-Register abgelegt. Ferner enthält die Registerkombination BL:CX eine 24-Bit-Seriennummer und das BH-Register gibt über die Registriernummer des OEM-Kunden Aufschluß.

## Händlernachweis

**1000** Berlin 30, Conrad Electronic, 0 30 / 2 61 70 59, **1000** Berlin-Neukölln, Hertie, 0 30 / 68 99 40, **1000** Berlin-Wilmersdorf, 0 30 / 31 03 11, **1000** Berlin-Spandau, Hertie, 0 30 / 33 30 92, **1000** Berlin 30, KaDeWe, 0 30 / 21 21-0, **1000** Berlin, Karstadt, Hermannplatz, 0 30 / 79 00 10, **1000** Berlin 28, WAW-Electronic, Tengelerstr. 2, 0 30 / 4 04 33 31, **1000** Berlin 30, Wegert GmbH & Co. KG, 0 30 / 2 50 02-0, **1000** Berlin-Kudamm, Wertheim, 0 30 / 3 81 90 71, **1000** Berlin-Steglitz, Wertheim, 0 30 / 79 10 91, **2000** Hamburg 36, Alsterhaus, 0 40 / 3 59 01-1, **2000** Hamburg 76, Conrad Electronic, 0 40 / 29 17 21, **2000** Hamburg 1, Karstadt, Mönckebergstr. 0 40 / 30 94-1, **2000** Hamburg 54, L & S Schaulandt GmbH, 0 40 / 47 70 07, **2000** Hamburg, TVS GmbH, 0 40 / 2 50 05 13, **2050** Hamburg, Gerd Messner, 0 40 / 7 23 82 42, **2070** Ahrensburg, Nessler GmbH, Electronics, 0 41 02 / 5 04 03, **2072** Bargteheide, MSD Computer, 0 45 32 / 22 13 15, **2100** Hamburg 90, Schaulandt, Hamburg GmbH, 0 40 / 77 24 64, **2240** Heide, Lorenzen & Partner GmbH, 0 48 1 / 7 42 02, **2300** Kiel 1, HCL Home Computer Laden, 0 43 1 / 55 55 55, **2300** Kiel, Hertie, 0 43 1 / 67 30 66, **2300** Kiel, MCC GmbH, Christ, 0 43 1 / 56 70 41, **2300** Kiel, Schaulandt Kiel, GmbH i.Gr., 0 43 1 / 67 60 51, **2400** Lübeck, Schaulandt, Lübeck GmbH, 0 45 1 / 7 13 51, **2800** Bremen 1, Hellweg Centrum, 0 42 1 / 44 00 55, **2800** Bremen 1, PS-Data, 0 42 1 / 17 05 77, **2802** Ottersberg/Posthsn., Dodenhof GmbH & Co. KG, 0 42 97 / 34 33, **3000** Hannover 1, Karstadt, 0 51 1 / 31 04-1, **3000** Hannover, Saturn Hansa GmbH, 0 51 1 / 45 50 01, **3057** Neustadt, W+W Computer, 0 50 32 / 6 42 49, **3180** Wolfsburg, Hertie, 0 53 61 / 2 50 51, **3300** Braunschweig, Media Markt, 0 53 1 / 4 02 51-55, **3300** Braunschweig, Unirep Elektronik Service GmbH, 0 53 1 / 87 40 02, **3440** Eschwege, Rathgeber EDV, 0 56 51 / 30 05-5, **4300** Essen 1, Conrad Electronic, 0 20 1 / 23 80 73, **4300** Essen, Karstadt, 0 20 1 / 17 61, **4400** Münster, Basis Computer Systeme GmbH, 0 25 1 / 71 99 75, **4500** Osnabrück, Foto Eck, 0 54 1 / 5 33 39, **4504** Georgsmarienhütte, Dacor GmbH, Computer-Shop, 0 54 01 / 4 54 41, **4600** Dortmund 1, Dacor-Computertechnik, Reschke GmbH, 0 23 1 / 52 74 04, **4600** Dortmund, Karstadt, 0 23 1 / 54 39 82 22, **4600** Dortmund 1, Saturn Hansa, 0 23 1 / 183 90, **4770** Soest, Dacor Comp. Shop GmbH, 0 29 21 / 6 38 03, **4770** Soest, Dacor Comp. Technik, 0 29 21 / 126 82, **4800** Bielefeld, Horten, 0 52 1 / 5 83 51 **4830** Gütersloh, Hertie, 0 52 41 / 86 50, **4970** Bad Oeynhausen, Unger Lutz oHG, Hifi-Studio, 0 57 3 / 2 71 18, **5000** Köln, Braun GmbH, Büromaschinen, 0 22 1 / 219 171, **5000** Köln, Hertie, 0 22 1 / 23 36 41, **5000** Köln, Karstadt, 0 22 1 / 20 39-0, **5000** Köln, Saturn Hansa, 0 22 1 / 161 60, **5100** Aachen, Horten, 0 24 1 / 2 96 61, **5160** Düren, MRZ Personalcomp., 0 42 1 / 7 10 75, **5300** Bonn 1, Hertie, 0 22 8 / 7 12-0, **5440** Mayen, Busch Präzision, 0 26 51 / 24 06, **5500** Trier, Kaurisch GmbH, 0 65 1 / 20 98 61, **5500** Trier, Shop 64, 0 65 1 / 2 24 41, **5600** Wuppertal, Hifi Studio 9, 0 20 2 / 55 50 53, **5600** Wuppertal 1, HMS GmbH, 0 20 2 / 4 93 69 12-13, **5630** Remscheid, HMS GmbH, 0 21 91 / 4 10 77, **6000** Frankfurt-Zeil, Hertie, 0 69 / 29 86-1, **6000** Frankfurt/Bornheim, Saturn Hansa GmbH, 0 69 / 40 50 10, **6000** Frankfurt 1, Schaulandt Frankfurt i.Gr., 0 69 / 28 08 93, **6090** Rüsselsheim, Büro Emig GmbH, 0 61 42 / 6 10 71, **6100** Darmstadt, Karstadt, 0 61 51 / 109-1, **6120** Erbach, Herbig, 0 60 62 / 36 25, **6200** Wiesbaden 1, Dacor GmbH, 0 61 34 / 6 00-1, **6200** Wiesbaden, Hertie, 0 61 21 / 14 80, **6200** Wiesbaden, Schaulandt Wiesbaden GmbH i. Gr., 0 61 21 / 30 50 18, **6250** Worms, Pro Markt, 0 62 41 / 5 20 28, **6300** Gießen, Karstadt, 0 64 1 / 7 00 41, **6330** Wetzlar/Lahn, Dacor, 0 64 41 / 4 11-0, **6400** Fulda, Media Markt, 0 66 1 / 80 13, **6457** Maintal 1, Landolt Computer, 0 61 81 / 4 52 93, **6600** Saarbrücken, Kaurisch, 0 68 1 / 40 00 50, **6600** Saarbrücken, Shop 64, 0 68 1 / 39 76 77, **6630** Saarlouis, Shop 64, 0 68 31 / 4 84 33, **6650** Homburg, Shop 64, 0 68 41 / 6 33 33, **6670** St. Ingbert, Shop 64, 0 68 94 / 38 19 33, **6680** Neunkirchen, Shop 64, 0 68 21 / 2 47 17, **6700** Ludwigshafen, Phora-Wessendorf GmbH & Co., 0 62 1 / 519 2 91, **6730** Neustadt, Shop 64, 0 63 21 / 3 14 69, **6750** Kaiserslautern, Kaurisch, 0 63 1 / 160 52, **6750** Kaiserslautern, Shop 64, 0 63 1 / 6 09 96, **6800** Mannheim, Gauch + Sturm, 0 62 1 / 85 00 40, **6800** Mannheim, Karstadt, 0 62 1 / 168 31, **6800** Mannheim, Phora-Wessendorf GmbH & Co., 0 62 1 / 107 05 42, **6800** Mannheim, Promarkt, 0 62 1 / 46 62 20, **6900** Heidelberg, Dacor, 0 62 21 / 30 24 37, **6900** Heidelberg, Phora-Wessendorf GmbH & Co., 0 62 21 / 16 00 61, **6900** Heidelberg, Pro-TV-GmbH, Elektro-Hifi-Markt, 0 62 21 / 2 74 26, **6940** Weinheim, Dacor, 0 62 01 / 60 01 89, **7000** Stuttgart, Horten, 0 71 1 / 219 01, **7000** Stuttgart 1, Schreiber Elektronik GmbH, 0 71 1 / 22 70 90 / 99, **7200** Tuttlingen, Klaiber GmbH & Co. KG, 0 74 61 / 60 22, **7500** Karlsruhe, Hertie, 0 72 1 / 2 69 01-06, **7520** Bruchsal, Jöst HiFi Video Center, 0 72 51 / 10 30 91 / 92, **7600** Offenburg, Büromaschinen Mangei, 0 78 1 / 70 81-4, **7800** Freiburg, Hertie, 0 76 1 / 3 18 55, **7850** Lörrach, Hertie, 0 76 21 / 4 60 46-49, **7920** Heidenheim, Fa. Coss Bissinger, 0 73 21 / 4 50 72, **7990** Friedrichshafen 1, Reich Wilfried, Elektronik-Radio-TV, 0 75 41 / 71 22, **8000** München, Conrad Electronic, 0 89 / 59 21 28, **8000** München 60, Elektro Egger, 0 89 / 88 67 11, **8000** München, Hertie, 0 89 / 55 12-0, **8000** München, Karstadt, 0 89 / 130 21, **8000** München, Media Markt,Europa GmbH, 0 89 / 31 60 26, **8000** München, Saturn Hansa, 0 89 / 5 78 51, **8000** München 2, Seemüller, 0 89 / 59 42 81, **8032** Gräfelfing, Pro Markt, 0 89 / 85 10 43, **8070** Ingolstadt, Britt KG, Computerbörse, 08 41 / 3 27 49, **8070** Ingolstadt, Media Markt, 08 41 / 63 40, **8200** Rosenheim, Media Markt, 0 80 31 / 4 30 91, **8201** Großkarolinenfeld, Wassmann EDV GmbH, 0 80 31 / 5 91 91, **8221** Palling, DNS, 0 86 29 / 14 00, **8225** Traunreuth, Dutge, Computer, 0 86 69 / 52 37, **8300** Landshut, Media Markt, 08 71 / 7 42 14, **8360** Deggendorf, Technoland, 09 91 / 50 82, **8390** Passau, Donaudata GmbH, 08 51 / 3 34 58, **8390** Passau, Technoland, 08 51 / 60 41, **8395** Hauzenberg, SWS Computer Shop, 0 85 86 / 21 74, **8400** Regensburg, Asiba, 09 41 / 5 22 40, **8400** Regensburg, Computerladen GmbH, Steinmetz Karl, 09 41 / 4 82 99, **8400** Regensburg, Media Markt, 09 41 / 7 20 92, **8400** Regensburg, Moderne Computer Rebentrost GmbH, 09 41 / 70 24 30, **8400** Regensburg, Zacharias, Fotohaus, 09 41 / 56 06 66 + 5 23 23, **8430** Neumarkt, Grasenhiller, 0 91 81 / 57 66, **8452** Hirschau, Conrad Electronic, 0 96 22 / 3 01 11, **8463** Rötz/Opf., Computer Corner Rötz, 0 99 76 / 32 65 22, **8480** Weiden, Hösl electronic, 09 61 / 31 6 56, **8500** Nürnberg 70, Conrad Electronic, 09 11 / 26 32 80, **8580** Bayreuth, Media Markt, 09 21 / 8 27 01, **8580** Bayreuth, Schirmer + Voth, 09 21 / 5 43 26, **8600** Bamberg, Media Markt, 09 51 / 6 55 90, **8700** Würzburg, Duttenhofer GmbH & Co. KG, 09 31 / 50 12 15, **8750** Aschaffenburg, Media Markt, 0 60 21 / 2 17 93, **8758** Goldbach, Hugo Fernseh, 0 60 21 / 5 17 88, **8804** Dinkelsbühl, PM-Data-Discount, 0 98 51 / 62 68, **8883** Gundelfingen, Bissinger GmbH, Bürotechnik, 0 90 73 / 70 23, **8900** Augsburg, CS Computer-Service, 08 21 / 31 20 51, **8900** Augsburg, Karstadt, 08 21 / 31 53-1.




Um die MS-DOS-Versionsnummer zu lesen, müssen Sie in das AH-Register den Wert 30hex schreiben. Der Inhalt der Register DX, SI, DI, BP, CS, DS, SS, ES und des Kennzeichenregisters ändert sich durch den Funktionsaufruf nicht.

### Funktion 31hex: Programm beenden

Die Funktion 31hex (»Keep Process«) dient dazu, den Ablauf des aktuellen Programms zu unterbrechen und in das aufrufende Programm zurückzukehren. Das ist in vielen Fällen die Kommandoebene des Betriebssystems MS-DOS. Im Gegensatz zu anderen Funktionen stellt der PC den Arbeitsspeicher, den das beendete Programm belegt, nach dem Funktionsaufruf nicht zur freien Verfügung.

Um ein Programm über die Funktion 31hex zu beenden, muß das AH-Register den Wert 31hex enthalten. Im AL-Register legen Sie den Rückkehrcode ab, den Sie durch Aufruf der Funktion 4Dhex erhalten. Über das DX-Register teilen Sie der Funktion 31hex mit, wieviele 16-Byte-Paragraphen das aufgerufene Programm einschließlich des Programm-Segment-Präfix (PSP) belegt.

Die Funktion 32hex ist für MS-DOS reserviert.

### Funktion 33hex: Break-Flag abfragen/ändern

Im Normalfall überprüft MS-DOS das Drücken der Tastenkombination [Ctrl-C] nur in bestimmten Fällen. Das ist beispielsweise bei fast allen Ein- und Ausgaben der Standardgeräte der Fall.

Dieses Verfahren der Abfrage von [Ctrl-C] spart zwar viel Zeit, schränkt aber die Unterbrechungsmöglichkeiten des Anwenders ein. Wenn Sie den PC selbst bei Diskettenoperationen unterbrechen wollen (beispielsweise bei der Fehlersuche in Assemblerprogrammen), müssen Sie eine häufigere Abfrage der Tastenkombination [Ctrl-C] anfordern. Dies geschieht durch Setzen des Break-Flags.

Um den Zustand des Break-Flag zu ändern, müssen Sie vor dem Aufruf der Funktion 33hex den Wert 1 im AL-Register abspeichern. In diesem Fall gibt der Inhalt des DL-Registers an, ob das Break-Flag gesetzt (DL-Register auf 1) gesetzt oder zurückgesetzt (DL-Register auf 0) werden soll.

Nach Beendigung der Funktion 33hex enthalten alle Register einschließlich des Kennzeichenregisters ihre ursprünglichen Werte.

Möchten Sie dagegen den Zustand des Break-Flag lediglich abfragen, so müssen Sie in das AL-Register den Wert 0 schreiben. In diesem Fall gibt die Funktion 33hex den Status des Break-Flag im DL-Register zurück. Der Inhalt der Register AX, BX, CX, DH, SI, DI, BP, CS, DS, SS, ES und des Kennzeichenregisters ändert sich durch den Funktionsaufruf nicht.

Um die Funktion 33hex aufzurufen, muß das AH-Register den Wert 33hex enthalten.

Die Funktion 34hex ist für MS-DOS reserviert.

### Funktion 35hex: Interruptvektor lesen

Die Funktion 35hex liest den Inhalt eines Interruptvektors. Um Probleme zu vermeiden, sollten Sie Interruptvektoren niemals direkt im Speicher ändern oder lesen.

Wenn Sie die Adresse eines Interruptvektors mit der Funktion 25hex ändern wollen, ist es sinnvoll, die ursprüngliche Interruptadresse zuvor mit der Funktion 35hex zu lesen und für die spätere Verwendung abzuspeichern. Die Funktion gibt über die Registerkombination ES:BX den Inhalt des gewünschten Interruptvektors zurück.

Um einen Interruptvektor zu lesen, muß das AH-Register den Wert 35hex enthalten und das AL-Register die Nummer des Interruptvektors angeben. Der Inhalt der Register AH, CX, DX, SI, DI, BP, CS, DS, SS und des Kennzeichenregisters ändert sich durch den Funktionsaufruf nicht.

### Funktion 36hex: Kapazität des aktuellen Laufwerks lesen

Die Funktion 36hex ermittelt die Speicherkapazität eines bestimmten Laufwerks. Vor dem Aufruf dieser Funktion müssen Sie in das DL-Register die Laufwerksnummer schreiben. Dabei steht der Wert 0 für das Standardlaufwerk, der Wert 1 für das Laufwerk A, der Wert 2 für das Laufwerk B, und so weiter.

Die Funktion 36hex gibt die Anzahl der Sektoren pro Cluster im AX-Register, die Gesamtanzahl der Cluster im DX-Register und die Anzahl der Bytes pro Cluster im CX-Register zurück. Das BX-Register teilt zusätzlich mit, wie viele Cluster verfügbar sind.

Wenn die angegebene Laufwerksnummer ungültig ist, enthält das AX-Register nach dem Aufruf der Funktion 36hex den Wert FFFFhex.

Sie wählen die Funktion 36hex aus, indem Sie in das AH-Register den Wert 36hex schreiben. Der Inhalt der Register SI, DI, BP, CS, DS, SS, ES und des Kennzeichenregisters ändert sich durch den Funktionsaufruf nicht.

Die Funktion 37hex ist für MS-DOS reserviert.

(Markus Zietlow/ma)

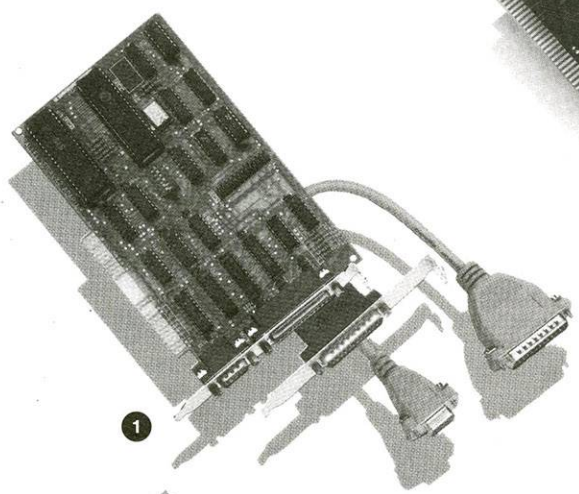
PROFEX

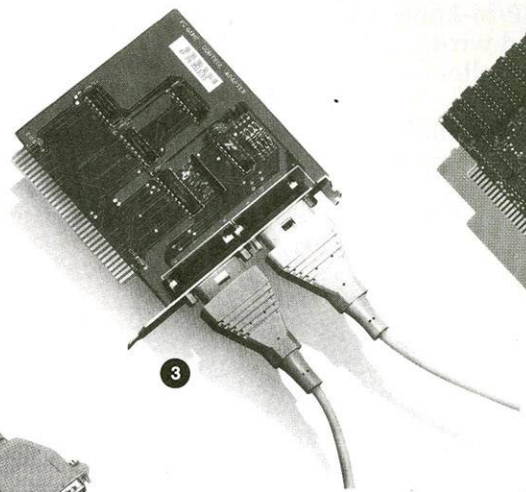
# PROFEX

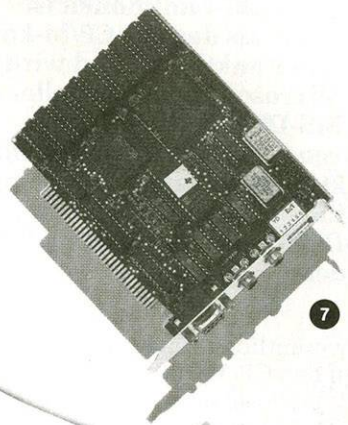
## zuverlässig und professionell

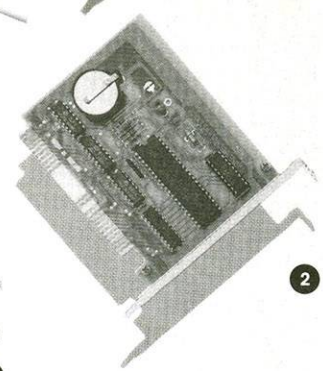
# ADD-ON-CARDS

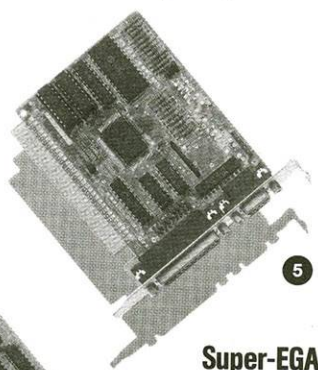
**ERWEITERUNGSKARTEN**
**für IBM*-kompatible Rechner.**

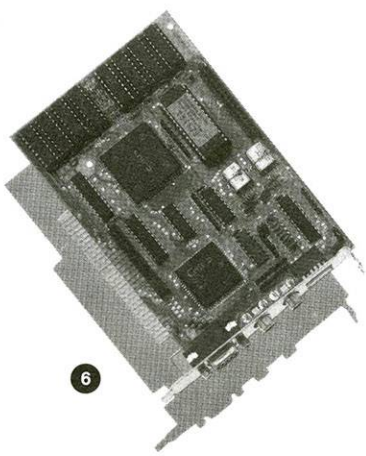


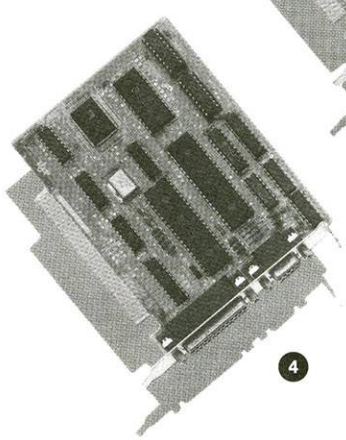




### Parallel/Serielle Schnittstelle für XT/AT ❶

enthält eine parallele und zwei serielle Schnittstellen. 2. serielle Schnittstelle (V 24) für separaten Slot sowie einen 9-Pin/V 24-Adapter.

### Echtzeituhr für XT ❷

Batteriegepufferte Echtzeituhr mit Datum.

### Game Card GC 1 für XT/AT ❸

Ermöglicht den Anschluß von 2 IBM*-kompatiblen Joysticks oder Paddles.

### Monochrom-Grafik-Karte für XT/AT ❹

Voll Hercules-kompatibel, MGA-Auflösung 720 x 348 monochrom, eine parallele Schnittstelle.

### Color-Grafik-Karte für XT/AT ❺

CGA-Auflösung 640 x 200 monochrom, 320 x 200 color, D 9 Steckeranschluß, eine parallele Schnittstelle, Lightpeninterface.

### Super-EGA-Level 3 Karte für XT/AT ❻

MDA-, CGA-, EGA-Auflösung 640 x 480 oder 640 x 350, PGA, analoges Ausgangssignal, Lightpeninterface, incl. Software.

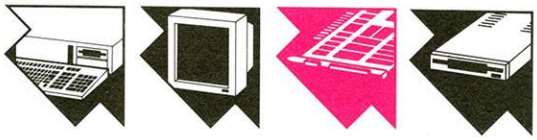
### Super-Hires-EGA-Level 5 Karte für XT/AT ❼

MDA-, CGA-, EGA-Auflösung 800 x 600 oder 640 x 480 oder 640 x 350 VGA, PGA, analoges Ausgangssignal, Lightpeninterface, incl. Software.

**ORGATECHNIK KÖLN '88**
INTERNATIONALE BÜROMESSE
**20. bis 25. Oktober**
Wir stellen aus: **Halle 3.1 / Stand N 52**

Händlerverzeichnis siehe linke Seite.

* IBM ist ein eingetr. Warenzeichen der International Business Machines Corp.



erhältlich im Fachhandel und in den Fachabteilungen der Warenhäuser.

PROFEX-Produkte im Vertrieb der **Electronic Handel KG,** 8391 Tiefenbach, Telefon (0 85 46) 19 - 177.

Vertrieb in der Schweiz durch Heutronic AG, CH-4601 Olten, Telefon 62 23 45 22

# Der MS-DOS-Interrupt 21hex (Teil 2)

## Die Xenix-kompatiblen Funktionen 38hex bis 62hex

**Das Konzept der Xenix-kompatiblen MS-DOS-Funktionen ist moderner als das der CP/M-kompatiblen Funktionen und wird von Microsoft, dem Hersteller von MS-DOS, fortgeführt, während die CP/M-kompatiblen Funktionen lediglich aus Kompatibilitätsgründen zu älteren MS-DOS-Versionen beibehalten werden.**

Der wesentliche Unterschied zwischen den CP/M- und Xenix-kompatiblen Funktionen besteht darin, daß die CP/M-kompatiblen Funktionen den Dateizugriff über einen Dateisteuerblock (FCB) vornehmen, während die Xenix-kompatiblen Funktionen den Zugriff auf eine Datei über einen sogenannten »Handle« steuern (ein Zugriffskanal für Dateien und Peripheriegeräte). Tabelle 1 zeigt, welche veralteten CP/M-kompatiblen Funktionen durch die moderneren Xenix-kompatiblen Funktionen ersetzt werden sollten.

Auch die Fehlermeldungen werden bei den CP/M- und Xenix-kompatiblen Funktionen unterschiedlich gehandhabt. Während bei den CP/M-kompatiblen Funktionen das AL-Register durch den Wert FFhex das Auftreten eines Fehlers anzeigt, signalisiert bei den Xenix-kompatiblen Funktionen das Carry-Flag des Kennzeichenregisters einen Fehler, und der Inhalt des AL-Registers gibt eine Nummer an, die die Art des Fehlers repräsentiert. Die Fehlermeldungen der Xenix-kompatiblen Funktionen sind in Tabelle 2 aufgeführt.

Tabelle 3 gibt schließlich die Parameter für die erweiterten Fehlermeldungen der Funktion 59hex und Tabelle 4 den Aufbau von DTA, FCB, PSP und so weiter an.

Im folgenden sind die Xenix-kompatiblen Funktionen des MS-DOS-Interrupt 21hex in der Reihenfolge ihrer Numerierung aufgeführt und werden mit Angabe der Ein- und Ausgabeparameter erläutert.

### Funktion38hex: Länderspezifische Daten ermitteln/setzen

Ein großer Vorteil des Betriebssystem MS-DOS ist, daß es sich relativ einfach an landesspezifische Eigenheiten wie beispielsweise die deutschen Umlaute anpassen läßt. Mit der Funktion 38hex lesen beziehungsweise setzen Sie die landesspezifischen Systemparameter von MS-DOS.

Wenn Sie in das DX-Register den Wert FFFFhex schreiben, können Sie mit der Funktion 38hex eine Landeskennzahl festlegen, die die landesspezifischen Parameter für MS-DOS angibt. Auf diese Weise läßt sich der PC auch noch nach dem Booten an landesspezifische Eigenheiten anpassen, was sonst nur über den Country-Befehl in der Datei »config.sys« möglich ist.

Zum Setzen der Landeskennzahl muß diese in das AL-Register geschrieben werden. Die Landeskennzahl entspricht in der Regel der internationalen Vorwahlnummer des jeweiligen Landes ohne die führenden Nullen. Beträgt der Wert des AL-Registers allerdings FFhex, so gibt nicht dieses Register, sondern das 16 Bit breite BX-Register eine erweiterte Landeskennzahl an.

Ist das AL-Register dagegen auf 0 gesetzt, so liest die Funktion 38hex die Landeskennzahl in das BX-Register und die landesspezifischen Parameter in einen Pufferbereich ein, dessen Startadresse von der Registerkombination DS:DX definiert wird.

In den Puffer werden 34 Byte geschrieben, die folgende Bedeutung haben: Byte 0 und 1 enthalten das Format für das Datum, Byte 2 bis 6 die Währungsbezeichnung des jeweiligen Landes, Byte 7 und 8 den ASCII-Wert des Tausender-Zeichens, Byte 9 und 10 das Dezimalpunkt-Zeichen, Byte 11 und 12 das Datum-Trennzeichen sowie Byte 13 und 14 das Trennzeichen der Zeitangabe.

Im 15. Byte befindet sich ein Bit-Feld, das das Währungsformat definiert. Das erste Bit bestimmt, ob das Währungssymbol vor (Bit 0 gleich 0) oder nach dem Wert (Bit 0 gleich 1) ausgegeben wird und das zweite Bit, ob das Betriebssystem zwischen Währungssymbol und Wert ein Leerzeichen einsetzt (Bit 1 gleich 1).

Byte 16 enthält die Anzahl der Stellen, die bei den Währungsangaben hinter dem Komma berücksichtigt werden, Byte 17 das Format der Zeitangabe (0: 12-Stunden-Uhr; 1: 24-Stunden-Uhr), Byte 18 bis 21 die Far-Adresse einer Routine, die das im AL-Register gespeicherte Zeichen in einen Großbuchstaben umwandelt sowie Byte 22 bis 23 das Daten-Trennzeichen. Byte 24 bis 33 sind reserviert.

Nachdem die Funktion 38hex beendet ist, gibt das Carry-Flag Aufschluß über den Erfolg der Aktion. Ist das Carry-Flag gesetzt, so war die angegebene Landeskennzahl ungültig. Hat die Funktion 38hex das Carry-Flag gelöscht, dann sind die der Landeskennzahl entsprechenden landesspezifischen Parameter erfolgreich eingestellt worden.

Sie wählen die Funktion 38hex an, indem Sie in das AH-Register den Wert 38hex schreiben. Der Inhalt der Register AX, BX, CX, DX, SI, DI, BP, CS, DS, SS und ES ändert sich durch den Funktionsaufruf nicht.

### Funktion 39hex: Anlegen eines Verzeichnisses

Mit der Funktion 39hex versuchen Sie, eine Datei zu erzeugen. Als Ergebnis erhalten Sie im Carry-Flag einen Wert, der Ihnen Aufschluß darüber gibt, ob die Funktion 39hex die

| Xenix-kompatible Funktion (hex) | CP/M-kompatible Funktion (hex) |
|---|---|
| Funktion 4C | Funktion 00 |
| Funktion 3D | Funktion 0F |
| Funktion 3E | Funktion 10 |
| Funktion 4E | Funktion 11 |
| Funktion 4F | Funktion 12 |
| Funktion 41 | Funktion 13 |
| Funktion 3F | Funktion 14 |
| Funktion 3D | Funktion 15 |
| Funktion 3C, 5A und 5B | Funktion 16 |
| Funktion 56 | Funktion 17 |
| Funktion 3F | Funktion 21 |
| Funktion 40 | Funktion 22 |
| Funktion 42 | Funktion 23 |
| Funktion 42 | Funktion 24 |
| Funktion 4B.00 | Funktion 26 |
| Funktion 3F | Funktion 27 |
| Funktion 40 | Funktion 28 |
| Funktion 4C | Interrupt 20 |
| Funktion 31 | Interrupt 27 |

**Tabelle 1. Diese Xenix-kompatiblen Funktionen sollten Sie anstelle der veralteten CP/M-Aufrufe benutzen.**

se Aktion erfolgreich abgeschlossen hat.

Ist auf dem aktuellen Datenträger genug Platz vorhanden, und ist das angegebene Oberverzeichnis existent, so löscht die Funktion 39hex das Carry-Flag und erzeugt das angegebene Unterverzeichnis. Konnte die Funktion das Unterverzeichnis dagegen nicht erzeugen, weil beispielsweise das Inhaltsverzeichnis vollgeschrieben ist, dann wird das Carry-Flag gesetzt. Das AX-Register enthält in diesem Fall den Fehlercode, dessen Bedeutung Sie der Tabelle 2 entnehmen können.

Um ein Verzeichnis anzulegen, muß das AH-Register den Wert 39hex enthalten und die Registerkombination DS:DX auf den Namen des Unterverzeichnisses zeigen, der mit einem Null-Byte abgeschlossen sein muß. Der Inhalt der Register BX, CX, DX, SI, DI, BP, CS, DS, SS, ES und des Kennzeichenregisters ändert sich durch den Funktionsaufruf nicht.

### Funktion 3Ahex: Unterverzeichnis löschen

Mit der Funktion 3Ahex löschen Sie ein Unterverzeichnis. Als Ergebnis erhalten Sie im Carry-Flag einen Wert, der Ihnen Aufschluß darüber gibt, ob die Funktion das Unterverzeichnis tatsächlich gelöscht hat.

Findet die Funktion 3Ahex zum Zeitpunkt des Aufrufs das angegebene Unterverzeichnis im aktuellen Inhaltverzeichnis, dann löscht sie das gewünschte Verzeichnis und setzt das Carry-Flag auf den Wert 0. Wenn sich das angegebene Unterverzeichnis aber nicht im aktuellen Verzeichnis befindet, setzt die Funktion 3Ahex das Carry-Flag und übergibt den Fehlercode in AX.

Um ein Verzeichnis zu löschen, muß das AH-Register den Wert 3Ahex enthalten und die Registerkombination DS:DX auf den Namen des Unterverzeichnisses zeigen, der mit einem Null-Byte abgeschlossen sein muß. Der Inhalt der Register BX, CX, DX, SI, DI, BP, CS, DS, SS, ES und des Kennzeichenregisters ändert sich durch den Funktionsaufruf nicht.

### Funktion 3Bhex: Wechseln des aktuellen Verzeichnisses

Mit der Funktion 3Bhex wechseln Sie das aktuelle Inhaltsverzeichnis. Als Ergebnis erhalten Sie im Carry-Flag einen Wert, der Sie über den Erfolg dieser Aktion informiert.

Ist das angegebene Verzeichnis vorhanden, so löscht die Funktion 3Bhex das Carry-Flag und wechselt das Verzeichnis. Konnte die Funktion das angegebene Verzeichnis dagegen nicht finden, dann setzt sie das

Die Fehlermeldungen der Xenix-kompatiblen Funktionsaufrufe (Carry = 1)

| AL (hex) | Fehlermeldung |
|---|---|
| 01 | Ungültiger Funktionsaufruf |
| 02 | Datei nicht gefunden |
| 03 | Pfad nicht gefunden |
| 04 | Zu viele Dateien geöffnet (kein Zugriffskanal mehr frei) |
| 05 | Unerlaubter Zugriff |
| 06 | Ungültiger Zugriffskanal |
| 907 | Speicher-Steuerblock zerstört |
| 08 | Freier Arbeitsspeicher zu klein |
| 09 | Ungültige Speicheradresse |
| 0A | Ungültige Betriebssystem-Umgebung |
| 0B | Ungültiges Dateiformat |
| 0C | Ungültige Zugriffsart |
| 0D | Ungültige Daten |
| 0E | Für MS-DOS reserviert |
| 0F | Ungültige Laufwerksangabe |
| 10 | Verzeichnis kann nicht gelöscht werden |
| 11 | Ungültiger Laufwerkswechsel |
| 12 | Keine weiteren Dateien vorhanden |
| 13 | Diskette schreibgeschützt |
| 14 | Fehler im Disketten-Controller |
| 15 | Laufwerk nicht bereit |
| 16 | Ungültiger Diskettenbefehl |
| 17 | Falsche Prüfsumme (CRC) |
| 18 | Ungültige Länge bei einer Diskettenoperation |
| 19 | Fehler beim Positionieren des Dateizeigers |
| 1A | Keine MS-DOS-Diskette |
| 1B | Sektor nicht gefunden |
| 1C | Kein Papier vorhanden |
| 1D | Schreibfehler |
| 1E | Lesefehler |
| 1F | Allgemeiner Fehler |
| 20 | Verletzung des gleichzeitigen Zugriffs auf einen Datensatz |
| 21 | Verletzung des nicht erlaubten Zugriffs auf einen gesperrten Datensatz |
| 22 | Falsche Diskette |
| 23 | Dateisteuerblock (FCB) nicht verfügbar |
| 24 bis 31 | Reserviert |
| 32 | Nicht netzwerkfähig |
| 33 | Gegenstelle empfängt nicht |
| 34 | Name tritt im Netzwerk doppelt auf |
| 35 | Name im Netzwerk nicht gefunden |
| 36 | Netzwerk aktiv |
| 37 | Netzwerktreiber nicht mehr vorhanden |
| 38 | Begrenzung einer BIOS-Funktion im Netzwerk überschritten |
| 39 | Hardwarefehler im Netzwerkadapter |
| 3A | Falsche Anforderung im Netzwerk |
| 3B | Unerwarteter Netzwerkfehler |
| 3C | Interface der Gegenstelle nicht kompatibel |
| 3D | Warteschlange des Druckpuffers voll |
| 3E | Warteschlange nicht voll |
| 3F | Nicht genug Platz vorhanden, um Datei auszudrucken |
| 40 | Name im Netzwerk wurde gelöscht |
| 41 | Zugriff nicht erlaubt |
| 42 | Typ des Netzwerktreibers falsch |
| 43 | Name im Netzwerk nicht gefunden |
| 44 | Begrenzung eines Netzwerknamens überschritten |
| 45 | Kurzzeitig angehalten |
| 46 | Netzwerkanforderung nicht akzeptiert |
| 47 | Umleitung auf Diskette oder Drucker wurde angehalten |
| 48 bis 4F | Reserviert |
| 50 | Datei bereits vorhanden |
| 51 | Reserviert |
| 52 | Befehl nicht ausführbar |
| 53 | Fehler im Interrupt 24h |
| 54 | Falsche Anordnung |
| 55 | Wurde bereits zugewiesen |
| 56 | Ungültiges Passwort |
| 57 | Ungültiger Parameter |
| 58 | Schreibfehler im Netzwerk |

**Tabelle 2. Der Interrupt 21hex schreibt den Fehlercode der Xenix-kompatiblen Funktionen in das AL-Register**

Carry-Flag, und das AL-Register enthält den Fehlercode.

Um das Verzeichnis zu wechseln, muß das AH-Register den Wert 3Bhex enthalten und die Registerkombination DS:DX auf den Namen des neuen Verzeichnisses zeigen, der mit einem Null-Byte abgeschlossen sein muß. Der Inhalt der Register BX, CX, DX, SI, DI, BP, CS, DS, SS, ES und des Kennzeichenregisters ändert sich durch den Funktionsaufruf nicht.

### Funktion 3Chex: Anlegen einer Datei über einen Handle

Die Xenix-kompatiblen Funktionen wickeln die Dateiverwaltung nicht über Dateisteuerblöcke (FCB), sondern über die sogenannten »Handles« ab. Ein Handle ist ein numerischer Wert, den das Betriebssystem einem bestimmten Ein- oder Ausgabegerät zuordnet. Der Zugriff auf diese Geräteeinheiten erfolgt ausschließlich über Handles.

Mit der Funktion 3Chex versuchen Sie, eine Datei zu erzeugen. Als Ergebnis erhalten Sie im Carry-Flag einen Wert, der Sie über den Erfolg dieses Versuches informiert.

Findet die Funktion 3Chex zum Zeitpunkt des Aufrufs den spezifizierten Eintrag im Inhaltverzeichnis, dann wird die Länge der bestehenden Datei auf 0 gesetzt. Existiert die Datei noch nicht, dann wird sie erzeugt. In beiden Fällen löscht die Funktion 3Chex das Carry-Flag und öffnet die Datei. Konnte die Funktion dagegen keine Datei erzeugen, weil beispielsweise das Inhaltsverzeichnis vollgeschrieben ist, dann setzt sie das Carry-Flag und übergibt die Fehlernummer im AX-Register.

Um eine Datei anzulegen, muß das AH-Register den Wert 3Chex enthalten und die Registerkombination DS:DX auf einen Dateieintrag zeigen, der mit einem Null-Byte endet. Das Register CX enthält vor dem Aufruf das Dateiattribut. Der Inhalt der Register BX, CX, DX, SI, DI, BP, CS, DS, SS, ES ändert sich durch den Funktionsaufruf nicht.

### Funktion 3Dhex: Öffnen einer bestehenden Datei über Handle

Mit der Funktion 3Dhex öffnen Sie eine existierende Datei. Findet die Funktion zum Zeitpunkt des Aufrufs den spezifizierten Eintrag im Inhaltsverzeichnis, so öffnet sie die Datei. In diesem Fall wird das Carry-Flag gelöscht und der Handle der Datei im AX-Register abgespeichert.

Konnte die Funktion 3Dhex die Datei dagegen nicht öffnen, beispielsweise weil sie nicht existiert, dann wird das Carry-Flag gesetzt und die Fehlernummer in AX übergeben.

Um eine existierende Datei zu öffnen, muß das AH-Register den Wert 3Dhex enthalten und die Registerkombination DS:DX auf den Dateieintrag zeigen, der mit einem Null-Byte endet.

Das Register AL enthält vor dem Aufruf den Zugriffscode. Die Bit 0 bis 3 des Zugriffscodes bestimmen die Zugriffsart. Die Datei darf entweder ausschließlich gelesen (Kombination 0000), nur beschrieben (Kombination 0001) oder gelesen und beschrieben werden (Kombination 0010). Die Bit 4 bis 6 des Zugriffscodes kontrollieren den Zugriff durch andere Programme. Der Zugriff auf die Datei ist entweder dem aktuellen Programm im FCB-Modus (Kombination 000) oder unter normalen Umständen vorbehalten (Kombination 001). Andere Programme dürfen die Datei entweder ausschließlich lesen (Kombination 010), nur beschreiben (Kombination 011) oder lesen und beschreiben (Kombination 100).

Der Inhalt der Register BX, CX, DX, SI, DI, BP, CS, DS, SS und ES ändert sich durch den Funktionsaufruf nicht.

### Funktion 3Ehex: Datei schließen über Handle

Mit der Funktion 3Ehex schließen Sie eine Datei. Ist zum Zeitpunkt des Funktionsaufrufs der angegebene Handle gültig, den die Funktion 3Ehex im CX-Register übernimmt, so löscht sie das Carry-Flag und schließt die Datei. Tritt dagegen ein Fehler auf, dann setzt die Funktion das Carry-Flag und legt den Fehlercode im AL-Register ab.

Um eine Datei zu schließen, muß das AH-Register den Wert 3Ehex enthalten. Der Inhalt der Register BX, CX, DX, SI, DI, BP, CS, DS, SS und ES ändert sich durch den Funktionsaufruf nicht.

Die Funktion 3Fhex liest Daten über einen im BX-Register angegebenen Handle ein und speichert sie in einem vorher definierten Eingabepuffer. Empfängt die Funktion Daten, obwohl der Puffer voll ist, dann gibt Sie eine Fehlermeldung weiter.

### Funktion 3Fhex: Zeichenkette über Handle einlesen

Wenn die Tastatur das Eingabemedium ist, dann können Sie die eingegebene Zeichenkette mit den dafür vor-

| Fehlerursache: | |
|---|---|
| BH (hex) | Bedeutung |
| 01 | Es steht kein Platz mehr auf dem Speichermedium zur Verfügung |
| 02 | Kurzfristiges Zugriffsverbot |
| 03 | Zugriff aufgrund mangelnder Autorisation nicht erlaubt |
| 04 | Interner Software-Fehler |
| 05 | Hardware-Problem |
| 06 | Fehler in der System-Software |
| 07 | Fehler im Anwendungsprogramm |
| 08 | Datei oder Einheit nicht gefunden |
| 09 | Datei oder Einheit ist falsch aufgebaut oder ungültig |
| 0A | Datei oder Einheit gesperrt |
| 0B | Es ist ein Problem mit Speichermedium aufgetreten |
| 0C | Unbekannter Fehler |
| **Fehlerquelle:** | |
| CH (hex) | Bedeutung |
| 01 | Nicht bekannt |
| 02 | Block-Treiber des Arbeitsspeichers |
| 03 | Netzwerk |
| 04 | Serielle Schnittstelle |
| 05 | Fehler im Arbeitsspeicher |
| **Ratschlag:** | |
| BL (hex) | Bedeutung |
| 01 | Vorgang wiederholen und bei weiterem Mißerfolg den Anwender befragen |
| 02 | Vorgang nach einer bestimmten Pause wiederholen |
| 03 | Anwender um nochmalige Eingabe der Daten bitten |
| 04 | Programm auf dem üblichen Weg beenden (inklusive Schließen der offenen Dateien) |
| 05 | Programm sofort beenden |
| 06 | Fehler ist unwichtig |
| 07 | Bitte an den Anwender, die Fehlerursache zu beseitigen und die Funktion noch einmal aufzurufen |

**Tabelle 3. Die Parameter für die erweiterten Fehlermeldungen der Funktion 59hex.**

Die Parameter und Parameterblöcke des MS-DOS-Interrupt 21

| Begriff | Bedeutung |
|---|---|
| Dateiattribut | Zeichen, das den Status einer Datei angibt<br>Attribut = 0: unbeschränkter Zugriff<br>Attribut = 1: Datei kann nur gelesen werden (Schreibschutz-Statusbit)<br>Attribut = 2: Verborgene Datei (Öffnen durch Funktion 0F nicht möglich)<br>Attribut = 4: Systemdatei (Öffnen durch Funktion 0F nicht möglich)<br>Attribut = 8: Hauptverzeichnis (Ändern durch Funktion 43 nicht möglich)<br>Attribut = 10: Unterverzeichnis (Ändern durch Funktion 43 nicht möglich)<br>Attribut = 20: Datei wurde beschrieben (Archiv-Statusbit)<br>Attribut = 40: Für MS-DOS reserviert<br>Attribut = 80: Für MS-DOS reserviert |
| Dateieintrag | Byte 00 bis 07: Dateiname (bei Bedarf mit Leerzeichen aufgefüllt)<br>Byte 08 bis 0A: Dateityp (bei Bedarf mit Leerzeichen aufgefüllt)<br>Byte 0B: Dateiattribut<br>Byte 0C bis 15: Für MS-DOS reserviert<br>Byte 16 und 17: Uhrzeit des letzten Schreibzugriffes<br>Byte 18 und 19: Datum des letzten Schreibzugriffes<br>Byte 1A und 1B: Erste belegte Datengruppe auf der Diskette/Festplatte<br>Byte 1C und 1D: Dateigröße = (Wort in 1C und 1D) + (Wort in 1E und 1F)x10<br>Byte 1E und 1F: Dateigröße = (Wort in 1E und 1F)x10 + (Wort in 1C und 1D) |
| DTA | Disketten – Übertragungsbereich (disk transfer area) für die Ein-/Ausgabe von Daten auf und von Diskette/Festplatte<br>Byte 00 bis 14: Für MS-DOS reserviert<br>Byte 15: Dateiattribut<br>Byte 16 und 17: Uhrzeit des letzten Schreibzugriffs<br>Byte 18 und 19: Datum des letzten Schreibzugriffs<br>Byte 1A und 1B: Dateigröße = (Wort in 1A und 1B) + (Wort in 1C und 1D)x10<br>Byte 1C und 1D: Dateigröße = (Wort in 1C und 1D)x10 + (Wort in 1A und 1B)<br>Byte 1E bis 2A: Dateiname<br>Byte 2B bis 7F: Pufferbereich |
| FAT | Dateibelegungstabelle (file allocation table) für die Zuordnung der Dateien auf Diskette/Festplatte zu den Angaben im Inhaltsverzeichnis des Datenträgers |
| FCB | Dateisteuerblock (file control block), der bei den CP/M-kompatiblen Funktionen den Zugriff auf eine Datei regelt<br>Folgende Angaben sind nur in einem erweiterten FCB vorhanden:<br>Byte -07 = FF: Kennzeichen für einen erweiterten FCB (Datei mit speziellem Dateiattribut)<br>Byte -06 bis -02: Für MS-DOS reserviert<br>Byte -01: Dateiattribut<br>Folgende Angaben sind in jedem FCB (auch ungeöffnet) enthalten:<br>Byte 00: Nummer des Laufwerks (0 = angewähltes Laufwerk, 1 = Laufwerk A und so weiter)<br>Byte 01 bis 08: Dateiname (bei Bedarf mit Leerzeichen aufgefüllt)<br>Byte 09 bis 0B: Dateityp (bei Bedarf mit Leerzeichen aufgefüllt)<br>Folgende Angaben sind (durch Funktion 0F) nur in einem geöffneten FCB vorhanden:<br>Byte 0C und 0D: Aktuelle Datenblocknummer (Voreinstellung: 0)<br>Byte 0E und 0F: Datenblockgröße (Voreinstellung: 80)<br>Byte 10 und 11: Dateigröße (Register)<br>Byte 12 und 13: Dateigröße (Segment)<br>Byte 14 und 15: Datum des letzten Schreibzugriffs<br>Bit 0 bis 4: Tag<br>Bit 5 bis 8: Monat<br>Bit 9 bis F: Jahr (0 = 1980, 1 = 1981, und so weiter)<br>Byte 16 und 17: Uhrzeit des letzten Schreibzugriffes<br>Bit 0 bis 4: Sekunden (1 = 2 Sekunden, 2 = 4 Sekunden, und so weiter)<br>Bit 5 bis A: Minuten<br>Bit B bis F: Stunden<br>Byte 18 bis 1F: Für MS-DOS reserviert<br>Byte 20: Aktueller Datensatz, sequentiell<br>Byte 21 und 22: Aktueller Datensatz, wahlfrei (Register)<br>Byte 23 und 24: Aktueller Datensatz, wahlfrei (Segment)<br>Byte 25 bis 27: Nicht verwendet<br>**(Die Registerkombination DS:DX enthält nicht die Startadresse, sondern jeweils den Zeiger auf den Anfang eines FCB)** |
| Handle | 16-Bit-Dateikennsatz zur Identifikation einer Datei beziehungsweise Zugriffskanal auf eine Datei (file handle), ersetzt das veraltete FCB-Verfahren<br>Folgende Handles sind bereits für Geräte vergeben:<br>Handle = 0: Standardeingabe CON (Tastatur)<br>Handle = 1: Standardausgabe CON (Bildschirm)<br>Handle = 2: Gerät, das für die Fehlerausgabe zuständig ist<br>Handle = 3: Ein-/Ausgabe AUX (serielle Schnittstelle)<br>Handle = 4: Druckerausgabe PRN (parallele Schnittstelle) |
| IOCTL | Kontrollfunktion (device I/O-control) für Peripheriegeräte (Funktion 44hex) |
| Kanalcode | Code für den Zugriff auf ein Gerät oder eine Datei |
| (Funktion 44hex) | Gerät:<br>Bit 0 = 1: Gerät ist Standardeingabe (CON)<br>Bit 1 = 1: Gerät ist Standardausgabe (CON)<br>Bit 2 = 1: Gerät ist Pseudoeinheit (NUL)<br>Bit 3 = 1: Gerät ist Systemuhr<br>Bit 4: Für MS-DOS reserviert<br>Bit 5 = 0: Control-Zeichen werden verarbeitet<br>Bit 5 = 1: Control-Zeichen [Ctrl-C], [Ctrl-P], [Ctrl-S] und [Ctrl-Z] werden nicht verarbeitet ([Ctrl-C] muß zusätzlich über Break oder Funktion 33 deaktiviert werden)<br>Bit 6 = 0: EOF erreicht<br>Bit 7 = 1: Kanalcode spezifiert Gerät<br>Bit 8 bis D: Für MS-DOS reserviert<br>Bit E = 1: Gerät kann Steuerdaten empfangen und senden<br>Bit F: Für MS-DOS reserviert<br>Datei:<br>Bit 0 bis 5: Angesprochenes Laufwerk (0 = A, 1 = B und so weiter)<br>Bit 6 = 0: Kanal wurde beschrieben<br>Bit 7 = 0: Kanalcode spezifiziert Datei<br>Bit 8 bis F: Für MS-DOS reserviert |
| PSP | Programmsegmentvorsatz (program segment prefix), der vom Betriebssystem ab Segmentadresse 100 vor jeder FCB-Datei im Arbeitsspeicher installiert wird und Informationen über die Behandlung der Datei enthält<br>Dieser Bereich darf von keinem Anwenderprogramm verändert werden:<br>Byte 00 und 01: Adresse des MS-DOS-Interrupt 20 (Programm beenden)<br>Byte 02 und 03: Erstes freies Byte hinter der Datei<br>Byte 04: Reserviert<br>Byte 05: Intersegment-Sprungbefehl<br>Byte 06 bis 09: Intersegmentadresse (IP:CS) zum Aufruf der MS-DOS-Funktionen 0 bis 24 (CL = Funktion)<br>Byte 0A bis 0D: Intersegmentadresse (IP:CS), von der aus diese Datei aufgerufen wurde<br>Byte 0E bis 11: Intersegmentadresse (IP:CS), die bei Auftreten von [Ctrl-C] aufgerufen wird<br>Byte 12 bis 15: Intersegmentadresse (IP:CS), die bei Auftreten eines Lesefehlers aufgerufen wird<br>Byte 16 bis 2B: Für MS-DOS reserviert<br>Byte 2C und 2D: Segmentadresse der Betriebssystem-Umgebung<br>Byte 2E bis 4F: Für MS-DOS reserviert<br>Byte 50 und 51: Adresse des MS-DOS-Interrupt 21 (Aufruf einer MS-DOS-Funktion)<br>Byte 52: RET (Intersegment)<br>Byte 53 bis 5B: Für MS-DOS reserviert<br>Dieser Bereich kann von einem Anwenderprogramm verändert werden:<br>Byte 5C bis 67: Ungeöffneter FCB für den ersten im Dateiaufruf übergebenen Dateinamen<br>Byte 68 bis 6B: Nicht verwendet<br>Byte 6C bis 77: Ungeöffneter FCB des zweiten im Dateiaufruf übergebenen Dateinamens<br>Byte 78 bis 7F: Nicht verwendet<br>Byte 80 bis FF: DTA (Ein-/Ausgabepuffer für Disketten- und Festplatten-Operationen) |
| Statuscode | Statuscode = 0: Programm korrekt beendet (Funktion 4C und 4Dhex)<br>Statuscode = 1: Programm durch [Ctrl-C] beendet<br>Statuscode = 2: Programm durch schweren Gerätefehler beendet<br>Statuscode = 3: Programm durch Funktion 31 beendet |
| Suchcode | Code zur Behandlung einer Dateibezeichnung, nach der gesucht werden soll (Funktion 29hex)<br>Bit 0 = 0: Abbrechen bei Auftreten eines Separators (»:« oder »\«)<br>Bit 0 = 1: Separatoren vor Dateinamen ignorieren<br>Bit 1 = 0: Wenn keine Laufwerksangabe vorhanden ist, »A« in FCB eintragen<br>Bit 1 = 1: Wenn keine Laufwerksangabe vorhanden ist, FCB unverändert lassen<br>Bit 2 = 0: Wenn kein Dateiname vorhanden ist, acht Leerzeichen in FCB eintragen<br>Bit 2 = 1: Wenn kein Dateiname vorhanden ist, FCB unverändert lassen<br>Bit 3 = 0: Wenn keine Dateitypangabe vorhanden ist, drei Leerzeichen in FCB eintragen<br>Bit 3 = 1: Wenn keine Dateitypangabe vorhanden ist, FCB unverändert lassen |

**Tabelle 4. Die Parameter der Funktionen des MS-DOS-Interrupt 21hex mit Bedeutung und internem Aufbau**

gesehenen Sondertasten editieren. Die Position des Eingabepuffers übergeben Sie mit dem Zeiger DS:DX an die Funktion.

Um eine Zeichenkette einzulesen, muß das AH-Register den Wert 3Fhex enthalten. Nach Bearbeitung der Funktion enthält der Eingabepuffer die eingegebene Zeichenkette. Vor dem Aufruf der Funktion 3Fhex geben Sie im CX-Register an, wie viele Bytes die Funktion in den Tastaturpuffer einlesen und speichern soll.

Der Inhalt der Register BX, CX, DX, SI, DI, BP, CS, DS, SS und ES ändert sich durch den Funktionsaufruf nicht.

Auch bei dieser Funktion enthält das AX-Register den Fehlercode, wenn die Funktion einen Fehler registriert und das Carry-Flag gesetzt hat.

**Funktion 40hex: Zeichenkette über Handle ausgeben**

Die Funktion 40hex dient dazu, Daten über einen im BX-Register angegebenen Handle auszugeben, die in einem vorher definierten Ausgabepuffer gespeichert sind. Wenn der Zugriff nicht erlaubt ist oder ein ungültiger Handle angegeben wurde, gibt die Funktion 40hex eine Fehlermeldung zurück. Die Schreiboperation beginnt dabei an der aktuellen Position des Dateizeigers, der nach dem Schreibvorgang aktualisiert wird. Die Position des Tastaturpuffers übergeben Sie mit dem Zeiger DS:DX an die Funktion.

Um eine Zeichenkette auszugeben, muß das AH-Register den Wert 40hex enthalten. Vor dem Aufruf der Funktion 40hex geben Sie im CX-Register an, wie viele Bytes die Funktion vom Tastaturpuffer über den Handle ausgeben soll. Der Inhalt der Register BX, CX, DX, SI, DI, BP, CS, DS, SS und ES ändert sich durch den Funktionsaufruf nicht.

**Funktion 41hex: Datei über Handle löschen**

Mit der Funktion 41hex löschen Sie eine Datei über den Handle. Findet die Funktion zum Zeitpunkt des Aufrufs den Namen der Datei im Inhaltsverzeichnis, so löscht sie die Datei und das Carry-Flag. Sie können mit der Funktion 41hex allerdings keine Verzeichnisse löschen.

Wenn die Datei, die die Funktion 41hex löschen soll, nicht im aktuellen Verzeichnis vorhanden ist, dann wird das Carry-Flag gesetzt, und AX enthält den Fehlercode. Sie dürfen keine Wildcards im Dateinamen benutzen.

Um eine Datei zu löschen, muß das AH-Register den Wert 41hex enthalten und die Registerkombination DS:DX auf den Dateieintrag zeigen, der mit einem Null-Byte abgeschlossen ist. Der Inhalt der Register BX, CX, DX, SI, DI, BP, CS, DS, SS und ES ändert sich durch den Funktionsaufruf nicht.

**Funktion 42hex: Position des Dateizeigers über Handle bestimmen**

Die Funktion 42hex dient dazu, die Position des Dateizeigers für den Lese- beziehungsweise Schreibzugriff über einen Handle zu setzen. Die neue Position des Dateizeigers setzt die Funktion 42hex nicht absolut sondern relativ zu einer bestimmten Position. Der Wert im AL-Register bestimmt diese Position.

Enthält das AL-Register den Wert 0, so bezieht sich die Positionsangabe auf den Anfang der Datei, beträgt der Inhalt von AL dagegen 1, dann bezieht er sich auf die aktuelle Position des Zeigers. Wenn AL schließlich der Wert 2 zugewiesen wird, berechnet die Funktion 42hex die neue Position relativ zum Dateiende.

Unabhängig vom AL-Register übergibt die Funktion 42hex in der Registerkombination DX:AX die Position des Dateizeigers relativ zum Dateianfang.

Beim Aufruf der Funktion 42hex muß das AH-Register den Wert 42hex und die 32-Bit-Registerkombination CX:DX den gewünschten Offset enthalten. Der Inhalt der Register BX, CX, SI, DI, BP, CS, DS, SS, ES und des Kennzeichenregisters ändert sich durch den Funktionsaufruf nicht.

**Funktion 43hex: Attribut einer Datei ermitteln beziehungsweise setzen**

Die Funktion 43hex ist dafür zuständig, die Attribute einer Datei zu verwalten. Mit dieser Funktion kann der Anwender die Attribute einer Datei lesen und modifizieren. Wenn Sie im AL-Register den Wert 0 übergeben und die Funktion 43hex aufrufen, erhalten Sie im CX-Register einen Wert, der das Dateiattribut angibt.

Bit 1 beinhaltet die Information, ob eine Datei nur gelesen (Bit 0 gleich 1) oder auch beschrieben werden darf (Bit 0 gleich 0). Bit 2 bestimmt, ob die Datei versteckt ist (Bit 1 gleich 1) oder nicht (Bit 1 gleich 0). Bit 5 gibt an, ob eine Datei seit der letzten Archivierung verändert worden ist (Bit 5 gleich 1) oder nicht (Bit 5 gleich 0). Aus den Bit 3 und 4 erkennt man schließlich, ob die Datei ein Volume (Bit 3 gleich 1) oder ein Unterverzeichnis ist (Bit 4 gleich 1).

Der Inhalt der Register BX, DX, SI, DI, BP, CS, DS, SS und ES ändert sich durch den Funktionsaufruf nicht.

Die zweite Variante der Funktion 43hex setzen Sie ein, um das Attribut einer Datei auf einen speziellen Wert zu setzen. Zu diesem Zweck belegen Sie das AL-Register mit dem Wert 1 und schreiben in das CX-Register das neue Dateiattribut.

Liegen die übergebenen Parameter im erlaubten Wertebereich, so löscht die Funktion 43hex die alten Angaben, aktualisiert sie und löscht das Carry-Flag. Wenn die Parameter, die Sie an die Funktion übergeben, nicht gültig sind, wird das Carry-Flag gesetzt und AX beinhaltet den Fehlercode. Der Inhalt der Register BX, CX, DX, SI, DI, BP, CS, DS, SS und ES ändert sich durch den Funktionsaufruf nicht.

Sie wählen die Funktion 43hex an, indem Sie in das AH-Register den Wert 43hex schreiben. In der Registerkombination DS:DX legen Sie zusätzlich den gewünschten Dateieintrag für die Datei ab, der mit einem Null-Byte endet.

**Funktion 44hex: Bearbeiten eines Gerätetreibers (IOCTL)**

Die Funktion 44hex verfügt über mehrere Unterfunktionen, die sich ausschließlich mit dem Zugriff auf die Gerätetreiber beschäftigen (IOCTL für »Input/Ouput Control«). Mit der Funktion 44hex kann der Anwender die Eigenschaften eines Treibers lesen, modifizieren und schreiben.

Wenn Sie im AL-Register den Wert 0 übergeben und die Funktion 44hex aufrufen, erhalten Sie im DX-Register den Kanalcode des Gerätetreibers.

Bit 0 beinhaltet die Information, ob der Treiber ein Tastaturtreiber ist (Bit 0 gleich 1). Bit 1 bestimmt, ob der Treiber ein Bildschirmtreiber ist (Bit 1 gleich 1) oder nicht (Bit 1 gleich 0). Bit 2 gibt an, ob es sich um einen Null-Treiber (Bit 5 gleich 1) oder um einen Uhr-Treiber (Bit 3 gleich 1) handelt. Ist Bit 5 gelöscht, dann reagiert der Treiber auf Control-Steuersequenzen. Bit 14 bestimmt, daß der Treiber über die Unterfunktionen 2 und 3 Steuersequenzen verarbeiten kann (Bit 14 gleich 1). Die Bit 4, 15 und 8 bis 13 sind reserviert.

Der Inhalt der Register BX, CX, SI, DI, BP, CS, DS, SS und ES ändert sich durch diesen Unterfunktionsaufruf nicht.

Die zweite Unterfunktion der Funktion 44hex rufen Sie auf, wenn das AL-Register den Wert 1 enthält. Diese Unterfunktion ist dafür zuständig, die Attribute des Gerätetreibers auf die vorgegebenen Parameter zu setzen. Im DX-Register übergeben Sie diese Parameter.

Liegen die Parameter im erlaubten Wertebereich, so löscht die Unterfunktion die alten Angaben, aktualisiert sie und löscht das Carry-Flag.

Sind die Parameter dagegen nicht gültig, dann wird das Carry-Flag gesetzt und das AL-Register beinhaltet den Fehlercode. Der Inhalt der Register BX, CX, DX, SI, DI, BP, CS, DS, SS und ES ändert sich durch diesen Unterfunktionsaufruf nicht.

Mit der dritten Unterfunktion (AL-Register auf 2 gesetzt) empfangen Sie eine bestimmte Anzahl Zeichen von einem Treiber. Die Unterfunktion übergibt die Zeichen in einem Puffer an das aufrufende Programm, der durch die Registerkombination DS:DX definiert ist.

Sendet der Treiber zum Zeitpunkt des Funktionsaufrufs kein Zeichen, so wartet die Unterfunktion.

Das CX-Register enthält die Anzahl der zu empfangenden Zeichen. Der Inhalt der Register BX, CX, DX, SI, DI, BP, CS, DS, SS und ES ändert sich durch diesen Unterfunktionsaufruf nicht.

Die vierte Unterfunktion (AL-Register auf 3 gesetzt) benutzen Sie, um eine bestimmte Anzahl Zeichen an einen Treiber zu senden. Das aufrufende Programm muß diese Zeichen in einem Puffer übergeben, der durch die Registerkombination DS:DX definiert wird. Empfängt zum Zeitpunkt des Funktionsaufrufs der Treiber kein Zeichen, dann wartet die Unterfunktion, bis der Treiber empfangsbereit ist.

Das CX-Register enthält die Anzahl der zu sendenden Zeichen. Der Inhalt der Register BX, CX, DX, SI, DI, BP, CS, DS, SS und ES ändert sich durch diesen Unterfunktionsaufruf nicht.

Die Unterfunktionen 5 und 6 basieren auf dem gleichen Prinzip wie die Unterfunktionen 3 und 4. Der einzige Unterschied besteht darin, daß die letztgenannten statt Blocktreiber Zeichentreiber bearbeiten. Die Parameterübergabe und -ausgabe ist aber identisch.

Mit den Unterfunktionen 7 und 8 lesen Sie den Ein- beziehungsweise Ausgabestatus eines Treibers. Das AX-Register enthält jeweils den Treiberstatus. Der ermittelte Wert beträgt 0, wenn der Treiber bereit ist, und FFhex, wenn der Treiber nicht bereit ist.

Wenn Sie herausfinden wollen, ob ein Laufwerk einen festen oder wechselbaren Datenträger hat, benutzen Sie die neunte Unterfunktion (AL-Register auf 8 gesetzt). In diesem Fall gibt das AH-Register an, ob der Datenträger wechselbar (AH gleich 0) oder fest installiert ist (AH gleich 1).

In Netzwerken sind die Unterfunktionen 10 und 11 von Bedeutung. Mit ihnen stellen Sie fest, ob ein angegebenes Laufwerk Teil des Computers ist, auf dem die Abfrage stattfindet, oder ob es zu einem Netzwerk gehört. Das DX-Register enthält den Status des Laufwerks. Bit 12 dieses Registers ist gesetzt, wenn das Laufwerk Teil eines Netzwerkes ist, anderenfalls ist es gelöscht.

Die zwölfte Unterfunktion (AL-Register auf 11 gesetzt) der Funktion 44hex bestimmt die Art des Zugriffs auf die Gerätetreiber und setzt die Anzahl der Wiederholversuche sowie die Länge der Pause zwischen den Zugriffsversuchen.

Gerade bei Netzwerken tritt oft der Fall ein, daß der Zugriff auf bestimmte Dateien für kurze Zeit gesperrt ist. In diesem Fall wiederholt das Betriebssystem seinen Zugriff mehrmals nach gewissen Pausen. Die Anzahl der Wiederholversuche geben Sie im BX-Register an. Das CX-Register defininiert die Pausen zwischen den Zugriffsversuchen. Dieser Wert ist modellabhängig, weil der Inhalt des CX-

**WENDIN-DOS** T.M. THE DOS OF THE FUTURE

* Jetzt mit deutschem Handbuch !

Das preisgünstige Multiuser-Betriebssystem für AT und 80386 oder kompatible Computer (mind 640 K).
Zusätzliche Terminals können Sie über die seriellen Schnittstellen anschließen.
Die meisten PC-DOS-Programme können Sie im Multiuser-Betrieb weiter benutzen.
Selbst wenn Sie nur mit einem Bildschirm arbeiten wollen, so können Sie nun gleichzeitig in einem Fenster editieren, während in einem anderen gerade ein Compiler arbeitet, und in einem dritten Dateien kopiert werden.

**Bestellen Sie das DOS der Zukunft noch heute für nur 298,-DM + Versand.**

**Das Application Developers Kit für C- und Assembler-Programmierer gibt's für 278,-DM.**

**POWER C**
Einer der mächtigsten C-Compiler.
Über 420 Funktionen incl. Grafik.
Handbuch mit über 650 Seiten incl. Tutorial. **79,80 DM.**

**POWER Ctrace Debugger**
Komplette Ablaufkontrolle in 7 Fenstern. **79,80 DM**

**Split-Screen Editor**
Editieren von 2 Dateien gleichzeitig. **79,80 DM**

**POWER C Business Math**
Mathematische Funktionen für Power C. **39,90 DM**

**POWER C Library Source Code**
Der Quell-Code der Power C Funktionen. **39,90 DM**

Alle Preise zuzüglich Versandkosten.

fecho software
Privileged Dealer Wendin Inc

WESELER STRASSE 117 - 4242 REES 1
TELEFON 0 28 51 / 12 19

Registers nicht eine feste Wartezeit, sondern eine Anzahl von Leerschleifen bestimmt.
Sie wählen die Funktion 44hex an, indem Sie in das AH-Register den Wert 44hex und in das AL-Register die Nummer der Unterfunktion (von 0 beginnend) schreiben. Bei den Unterfunktionen mit den AL-Registerwerten 0, 1, 2, 3, 6, 7 und 10 speichern Sie vor dem Aufruf den Handle im BX-Register ab. Die Unterfunktionen 4, 5, 8 und 9 benötigen die Laufwerksnummer im BL-Register. Dabei steht der Wert 1 für das Laufwerk A, der Wert 2 für das Laufwerk B, der Wert 3 für die Festplatte C, und so weiter.

**Funktion 45hex: Datei-Handle verdoppeln**

Beim Aufruf der Funktion 45hex richtet MS-DOS für den im BX-Register übergebenen Handle einen zweiten Handle ein, der im AX-Register zurückgegeben wird. Der neue Handle hat die gleichen Eigenschaften wie der übergebene Handle. Sogar die Position des Dateizeigers stimmt überein.
Zum Aufruf der Funktion 45hex muß das AH-Register den Wert 45hex enthalten. Der Inhalt der Register BX, CX, DX, SI, DI, BP, CS, DS, SS und ES ändert sich durch den Funktionsaufruf nicht.

**Funktion 46hex: Zwei Datei-Handles angleichen**

Beim Aufruf der Funktion 46hex gleicht MS-DOS den im CX-Register enthaltenen Handle an den im BX-Register übergebenen Handle an. Der im CX-Register übergebene Handle hat nach Ausführen der Funktion 46hex die gleichen Eigenschaften wie der Handle im BX-Register. Sogar die Position des Dateizeigers stimmt überein. Bearbeitet der zweite Handle eine geöffnete Datei, dann wird diese vor dem Vorgang der Angleichung geschlossen.
Zum Aufruf der Funktion 46hex muß das AH-Register den Wert 46hex enthalten. Der Inhalt der Register BX, CX, DX, SI, DI, BP, CS, DS, SS und ES ändert sich durch den Funktionsaufruf nicht.

**Funktion 47hex: Lesen des aktuellen Verzeichnisses**

Mit der Funktion 47hex lesen Sie das aktuelle Inhaltsverzeichnis des im DL-Register angegebenen Laufwerks. Dabei steht der Wert 0 für das Standardlaufwerk, der Wert 1 für das Laufwerk A, der Wert 2 für das Laufwerk B, und so weiter. Konnte die Routine das Unterverzeichnis nicht lesen, weil es beispielsweise das angegebene Laufwerk nicht gibt, dann setzt sie das Carry-Flag, und das AL-Register enthält den Fehlercode.
Zum Aufruf der Funktion 47hex muß das AH-Register den Wert 47hex enthalten und die Registerkombination DS:SI auf einen 64 Byte großen Puffer zeigen, in dem die Funktion 47hex den Verzeichnisnamen ablegt. Der Inhalt der Register BX, CX, DX, SI, DI, BP, CS, DS, SS und ES ändert sich durch den Funktionsaufruf nicht.

**Funktion 48hex: Arbeitsspeicher reservieren**

Die Funktion 48hex (»Allocate Memory«) dient dazu, Speicherbereich für die Verwaltung eines Programms zur Verfügung zu stellen. Als Ergebnis erhalten Sie im AX-Register die Segmentadresse des benötigten Arbeitsspeichers. Der reservierte Bereich beginnt ab der Adresse AX:0000hex.
Um Arbeitsspeicher zu reservieren, muß das AH-Register den Wert 48hex enthalten. Über das BX-Register teilen Sie der Funktion 48hex mit, wie viele 16-Byte-Paragraphen benötigt werden. Nachdem die Funktion 48hex ausgeführt ist, enthält das BX-Register die Anzahl der zur Verfügung stehenden Paragraphen. Der Inhalt der Register CX, DX, SI, DI, BP, CS, DS, SS und ES ändert sich durch den Funktionsaufruf nicht.

**Funktion 49hex: Arbeitsspeicher freigeben**

Die Funktion 49hex gibt Arbeitsspeicher, den das Betriebssystem zuvor mit der Funktion 48hex reserviert hat, wieder frei. Um Arbeitsspeicher freizugeben, muß das AH-Register den Wert 49hex enthalten. Über das ES-Register teilen Sie der Funktion 49hex mit, wo das Speichersegment des reservierten Speicherbereichs beginnt.
Der Inhalt der Register BX, CX, DX, SI, DI, BP, CS, DS, SS und ES ändert sich durch den Funktionsaufruf nicht.

**Funktion 4Ahex: Größe eines reservierten Speicherbereichs ändern**

Die Funktion 4Ahex (»Set Block«) dient dazu, den für die Verwaltung eines Programms zur Verfügung gestellten Speicherbereich zu ändern. Der reservierte Bereich beginnt an der alten Startadresse.
Um die Funktion 4Ahex aufzurufen, muß das AH-Register den Wert 4Ahex enthalten. Im BX-Register teilen Sie der Funktion mit, wie viele 16-Byte-Paragraphen die Funktion 4Ahex insgesamt zur Verfügung stellen soll, und im ES-Register steht die Segmentadresse des Speicherbereichs.
Nachdem die Funktion 4Ahex ausgeführt ist, enthält das BX-Register die Anzahl der zur Verfügung stehenden Paragraphen. Der Inhalt der Register CX, DX, SI, DI, BP, CS, DS, SS und ES ändert sich durch den Funktionsaufruf nicht.

**Funktion 4Bhex: Programm in den Arbeitsspeicher laden beziehungsweise zusätzlich ausführen**

Die Funktion 4Bhex ist dafür zuständig, ein Programm in den Speicher zu laden. Wenn Sie zusätzlich im AL-Register den Wert 0 übergeben und die Funktion aufrufen, wird das Programm auch gestartet. Ein übergebener Parameterblock, auf den die Registerkombination ES:BX weist, kontrolliert das Laden und Ausführen des Programms.
Die Byte 0 und 1 enthalten die Segmentadresse eines Environment-Blocks, der beispielsweise den Status des Verify-Flags oder des Break-Flags enthalten kann. Die Kombination der Byte 4 bis 5 (Segmentadresse) und 2 bis 3 (Offsetadresse) bestimmt die Adresse der Kommandozeile, die nicht länger als 128 Byte sein darf und mit einem ASCII-Zeichen 13 enden muß. Das erste Byte der Kommandozeile gibt ihre eigene Länge in Byte an, wobei das Byte zur Längenangabe mitgezählt und das abschließende ASCII-Zeichen 13 ignoriert wird.
Die Byte 6 bis 7 (Segmentadresse) und 8 bis 9 (Offsetadresse) geben die Adresse des ersten Dateisteuerblocks (FCB) an. Die Byte 10 bis 11 (Segmentadresse) und 12 bis 13 (Offsetadresse) definieren schließlich die Adresse des zweiten FCB.
Schreiben Sie in das AL-Register statt der 0 den Wert 3, so lädt die Funktion 4Bhex ein Programm in den Speicher, startet es aber nicht.
In den Registern übergeben Sie dabei die gleichen Parameter wie bei der ersten Variante. Der Inhalt der Register BX, CX, DX, SI, DI, BP, CS, DS, SS und ES ändert sich in beiden Fällen durch den Funktionsaufruf nicht.
Sie wählen die Funktion 4Bhex aus, indem Sie in das AH-Register den Wert 4Bhex schreiben. In der Registerkombination DS:DX speichern Sie zusätzlich einen Zeiger auf den Dateieintrag für den gewünschten Dateinamen ab, der mit einem Null-Byte endet und keine Wildcards enthalten darf.

**Funktion 4C: Programm beenden**

Die Funktion 4Chex (»End Process«) wird benutzt, um den Ablauf des aktuellen Programms zu unterbrechen und in das aufrufende Programm zurückzukehren. Letzteres ist meistens die Kommandoebene des Betriebs-

systems MS-DOS. Bevor die Funktion 4Chex die Kontrolle an das aufrufende Programm übergibt, restauriert sie die drei Interruptvektoren, deren Inhalt vor dem Start des Programms im Programm Segment Präfix (PSP) gespeichert wurden.

Der PC stellt den Arbeitsspeicher, den das beendete Programm belegt, nach dem Funktionsaufruf anderen Programmen zur Verfügung und schreibt den Inhalt aller Dateipuffer auf den angesprochenen Datenträger.

Um ein Programm zu beenden, muß das AH-Register den Wert 4Chex enthalten. Im AL-Register legen Sie den Statuscode, den Sie durch Aufruf der Funktion 4Dhex erhalten, und im CS-Register die Segmentadresse des zugehörigen PSP ab.

**Funktion 4Dhex: Statuscode ermitteln**

Mit der Funktion 4Dhex ermitteln Sie den Statuscode, den Sie für die Funktionen 31hex und 4Chex benötigen. Nachdem Sie die Funktion 4Dhex aufgerufen haben, enthält das AL-Register den Statuscode. Zusätzlich enthält das AH-Register einen Hinweis, warum ein Programm unterbrochen wird. Ist das AH-Register auf 0 gesetzt, so wurde das Programm regulär beendet. Der Wert 1 gibt dagegen an, daß die Ausführung des Programms durch die Tastenkombination [Control-C] unterbrochen wurde. Der Wert 2 zeigt an, daß ein Fehler bei dem Zugriff auf ein Gerät aufgetreten ist, und der Wert 3 bedeutet, daß das Programm durch den Aufruf der Funktion 00hex beendet wurde.
Um den Statuscode zu ermitteln, muß das AH-Register den Wert 4Dhex enthalten. Der Inhalt der Register BX, CX, DX, SI, DI, BP, CS, DS, SS und ES ändert sich durch den Funktionsaufruf nicht.

**Funktion 4Ehex: Ersten Eintrag im Inhaltsverzeichnis suchen**

Die Funktion 4Ehex sucht den ersten Eintrag im Inhaltsverzeichnis des aktuellen Verzeichnisses. Findet die Funktion zum Zeitpunkt des Aufrufs einen Eintrag im Inhaltverzeichnis, dann liest sie die Dateiinformationen ein. Wenn kein Eintrag vorhanden ist, dann wird das Carry-Flag gesetzt, und das AL-Register enthält den Fehlercode.
Die Funktion 4Ehex sucht die Datei mit dem im CX-Register angegebenen Attribut. Beim Aufruf dieser Funktion muß das AH-Register den Wert 4Ehex enthalten und die Registerkombination DS:DX auf den Dateieintrag zeigen, der mit einem Null-Byte endet und auch ein Verzeichnis oder Volume darstellen kann. Der Name kann aus Laufwerksangaben, Unterverzeichnissen und Wildcards bestehen. Der Inhalt der Register BX, CX, DX, SI, DI, BP, CS, DS, SS und ES ändert sich durch den Funktionsaufruf nicht.

**Funktion 4Fhex: Nächsten Eintrag im Inhaltsverzeichnis suchen**

Die Funktion 4Fhex sucht den nächsten Eintrag im aktuellen Inhaltsverzeichnis, nachdem mit der Funktion 4Ehex der erste Eintrag gefunden wurde. Findet die Funktion 4Fhex einen Eintrag, so wird das Carry-Flag gelöscht und die gewünschten Informationen werden eingelesen. Ist dagegen kein Eintrag vorhanden, dann setzt die Funktion das Carry-Flag und legt im AL-Register den Fehlercode ab.

Zum Aufruf der Funktion 4Fhex muß das AH-Register den Wert 4Fhex enthalten und die Registerkombination


# TEXTERKENNUNG + GRAFIKEINGABE

## ist nun mit dem bekannten HANDY-SCANNER möglich !

Hardware: Manueller Hand-Scanner, Scan-Breite 64mm, Auflösung 200 dpi (bei GCR: 200 mal 400 dpi). 512 Punkte/Zeile, schwarz/weiß. Geschwindigkeit 3ms/Zeile, Stromversorgung erfolgt durch den Computer. Interfacekarte für IBM-PC/XT/AT-kompatible Computer im Lieferumfang.

Grafikmodus: Bedienung mittels Maus oder Tastatur. Zwei Programme erlauben die bequeme Grafikeingabe: Mit dem mitgelieferten Treiber können Sie direkt in zahlreiche Standard-Pakete (z.B. PC-Paintbrush, Dr. Halo etc.) einscannen. Das eigenständige Programm "Grafik" erlaubt Ihnen das Erzeugen von MS-kompatiblen Grafikdateien. Eine ausführliche Bedienungsanleitung ergänzt das Paket.

Texteingabe: Das mitgelieferte Texterkennungs-Paket GCR (GERDES-CHARACTER-RECOGNITION) setzt neue Maßstäbe in Bezug auf das Preis/Leistungsverhältnis. Leistungsdaten des einzigartigen Leseverfahrens sind: Blockschrift und Proportionalschrift (sogar mit Kerning, d.h. Unterschneiden von Buchstaben!), schriftgrößenunabhängig (Mindestgröße: 10 Punkt), nahezu schriftartenunabhängig, lernfähig etc. Das mitgelieferte Anwendungsprogramm erlaubt Ihnen die Massentexteingabe. Dabei werden Standard-ASCII-Files erzeugt, die Sie mit praktisch jeder Textverarbeitung weiterbearbeiten können. Update-Service !

HANDY-SCANNER (komplett) mit GCR zum Sensationspreis von **DM 898.-**

Bestellungen gegen Vorkasse oder NN (+ DM 6,-) bei:

**Imperial SOFTware Systems Gerdes KG, Rochus-Center**

Lessenicher Straße 9, D-5300 Bonn 1, Telefon (0228) 616210 oder 252474
Dort erhalten Sie auch ein kostenloses Info (bitte Computertyp angeben).

DS:DX auf den Dateieintrag zeigen, der mit einem Null-Byte abgeschlossen ist. Der Inhalt der Register BX, CX, DX, SI, DI, BP, CS, DS, SS, ES und des Kennzeichenregisters ändert sich durch den Funktionsaufruf nicht.

Die Funktionen 50hex bis 53hex sind für MS-DOS reserviert.

**Funktion 54hex: Verify-Flag lesen**

Auf Wunsch überprüft MS-DOS alle Daten, die es auf einen Datenträger geschrieben hat, auf ihre Richtigkeit. Diese Vorgehensweise ist für die Datensicherheit zwar förderlich, benötigt aber zusätzliche Zeit. Benutzen Sie hochwertige Disketten, so können Sie getrost auf diese Überprüfung verzichten.

Die Funktion 54hex liest den Status des Verify-Flags. Wenn das Verify-Flag gesetzt ist, das heißt, wenn die geschriebenen Daten überprüft werden, speichert die Funktion 54hex den Wert 1 im AL-Register ab, andernfalls den Wert 0.

Um das Verify-Flag zu lesen, müssen Sie in das AH-Register den Wert 54hex schreiben. Nach der Rückkehr aus der Funktion enthalten alle Register einschließlich des Kennzeichenregisters mit Ausnahme des AL-Registers ihre ursprünglichen Werte.
Die Funktion 55hex ist für MS-DOS reserviert.

**Funktion 56hex: Datei über Handle umbenennen beziehungsweise verschieben**

Mit der Funktion 56hex benennen Sie eine oder mehrere Dateien um. Findet die Funktion 56hex zum Zeitpunkt des Aufrufs den Namen der zu ändernden Datei im Inhaltsverzeichnis, so löscht sie das Carry-Flag und übernimmt den neuen Namen für die Datei, indem sie den Verzeichniseintrag ändert.

Sie dürfen bei der Angabe des Dateinamens aber keine Wildcards benutzen und somit auch nicht den Namen einer Dateigruppe ändern. Wenn die Funktion 56hex die Datei nicht findet, oder der neue Name bereits existiert, wird das Carry-Flag auf den Wert 1 gesetzt.

Um eine Datei umzubenennen, muß das AH-Register den Wert 56hex enthalten und die Registerkombination DS:DX auf den alten sowie die Kombination ES:DI auf den neuen Dateieintrag zeigen. Die Dateieinträge müssen von einem Null-Byte abgeschlossen sein. Der Inhalt der Register BX, SI, DI, BP, CS, DS, SS und ES ändert sich durch den Funktionsaufruf nicht.

**Funktion 57hex: Zeit- und Datumsattribut einer Datei ermitteln beziehungsweise setzen**

Damit der Programmierer stets weiß, bei welcher Dateiversion es sich um die aktuellste handelt, speichert MS-DOS zu jedem Dateieintrag auch ein Zeit- und Datumsattribut ab, das den Zeitpunkt der letzten Änderung an der entsprechenden Datei angibt.

Mit der Funktion 57hex können Sie das Zeit- und Datumsattribut lesen und setzen. Wenn Sie in das AL-Register den Wert 0 schreiben und die Funktion 57hex aufrufen, erhalten Sie im CX-Register einen Wert, der das Zeitattribut angibt, und im DX-Register einen Wert, der das Datumsattribut anzeigt. Bit 0 bis 4 des CX-Registers enthalten die Sekunden, Bit 5 bis 10 die Minuten und Bit 11 bis 15 die Stunden der Zeitangabe. Bit 0 bis 4 des DX-Registers geben den Tag, Bit 5 bis 8 den Monat und Bit 9 bis 15 das Jahr relativ zu 1980 an.

Der Inhalt der Register BX, SI, DI, BP, CS, DS, SS und ES ändert sich durch diesen Funktionsaufruf nicht.

Um das Zeit- und Datumsattribut zu setzen, müssen Sie in das AL-Register den Wert 1 schreiben. Die Zeitangabe für das Zeitattribut übergeben Sie im CX-Register und die Datumsangabe für das Datumsattribut im DX-Register. Das Format für Zeit und Datum entspricht dem Format beim Lesen der beiden Attribute.

Sind die übergebenen Parameter gültig, so löscht die Funktion 57hex die alten Angaben, aktualisiert sie und löscht das Carry-Flag. Wenn die Parameter allerdings nicht korrekt sind, wird das Carry-Flag gesetzt, und das AL-Register enthält den Fehlercode. Der Inhalt der Register BX, CX, DX, SI, DI, BP, CS, DS, SS und ES ändert sich durch diesen Funktionsaufruf nicht.

Sie wählen die Funktion 57hex aus, indem Sie in das AH-Register den Wert 57hex schreiben. Im BX-Register legen Sie zusätzlich den gewünschten Handle des Gerätetreibers ab.

**Funktion 58hex: Konzept der Speicherzuteilung lesen beziehungsweise setzen**

MS-DOS teilt den aufgerufenen Programmen den jeweils benötigten Speicherplatz im Arbeitsspeicher zu. Es gibt grundsätzlich drei Konzepte, nach denen MS-DOS vorgeht. Beim ersten Verfahren durchsucht das Betriebssystem den Arbeitsspeicher von der niedrigsten Adresse bis zur höchsten und beendet die Suche, wenn es genügend freien Speicherplatz gefunden hat. Die zweite Strategie, die normalerweise Einsatz findet, sucht den freien, zusammenhängenden Bereich, dessen Größe am wenigsten über die angeforderte Speicherplatzlänge hinausgeht. Die dritte Methode durchsucht den Speicher wie das erste Verfahren – nur diesmal von oben nach unten.

Mit der ersten Unterfunktion der Funktion 58hex kann der Anwender das Konzept der verwendeten Speicherverwaltung ermitteln. Zu diesem Zweck müssen Sie in das AL-Register den Wert 0 schreiben, wenn Sie die Funktion 58hex aufrufen und erhalten im AX-Register die Nummer des angewandten Speicherkonzepts. Ist das AX-Register auf 0 gesetzt, so gilt das erste Konzept, beträgt der Inhalt 1, so wird das zweite Verfahren angewandt, und wird der Wert 2 zurückgegeben, so arbeitet MS-DOS nach der dritten Methode. Der Inhalt der Register BX, CX, DX, SI, DI, BP, CS, DS, SS und ES ändert sich durch diesen Unterfunktionsaufruf nicht.

Die zweite Unterfunktion der Funktion 58hex ist das Gegenstück zur ersten Unterfunktion, denn mit ihrer Hilfe können Sie das Speicherzuteilungskonzept selbst festsetzen. Zu diesem Zweck schreiben Sie in das AL-Register den Wert 1 und übergeben im BX-Register die Nummer des gewünschten Konzepts. Liegt der übergebenen Parameter im Wertebereich, dann löscht das Betriebssystem die alte Angabe, aktualisiert sie und löscht das Carry-Flag. Wenn der übergebene Wert dagegen ungültig ist, wird das Carry-Flag gesetzt, und das AL-Register enthält den Fehlercode. Der Inhalt der Register BX, CX, DX, SI, DI, BP, CS, DS, SS und ES ändert sich durch diesen Unterfunktionsaufruf nicht.

Sie wählen die Funktion 58hex an, indem Sie in das AH-Register den Wert 58hex schreiben.

**Funktion 59hex: Erweiterte Fehlermeldung holen**

Die Funktion 59hex erlaubt es Ihnen, detaillierte Informationen über einen Fehler zu erhalten, der beim Aufruf einer Funktion des Interrupt 21hex oder 24hex aufgetreten ist. Diese Angaben schließen nicht nur eine genaue Angabe der Fehlerursache und Fehlerquelle ein, sondern auch einen Ratschlag, welche Aktionen man zur Behebung der Fehlerursache durchführen sollte. Wenn Sie die Funktion 59hex aufrufen, muß das AH-Register den Wert 59hex enthalten. Nach der Rückkehr aus der Funktion enthält das AX-Register die Fehlerbeschreibung, das BH-Register die Fehlerursache, das CH-Register die Fehler-

quelle und das BH-Register die empfohlene Vorgehensweise. Der Inhalt der Register CS, SS und ES ändert sich durch den Funktionsaufruf nicht. Die genaue Bedeutung der Register AX, BH, BL und CH entnehmen Sie den Tabellen 2 und 3.

**Funktion 5Ahex: Anlegen einer temporären Datei über Handle**

Mit der Funktion 5Ahex erzeugen Sie eine temporäre Datei. Die Funktion 5Ahex erzeugt eine Datei, die von MS-DOS kontrolliert und zur zeitweisen Datenauslagerung benutzt wird. MS-DOS bestimmt den Namen dieser Datei selbst, so daß Sie auf die Angabe dieses Parameters verzichten können.

Zum Aufruf der Funktion 5Ahex muß das AH-Register den Wert 5Ahex enthalten und die Registerkombination DS:DX auf den Namen des Verzeichnisses zeigen, das die temporäre Datei aufnehmen soll. Der Verzeichnisname muß von einem Null-Byte beendet sein. Das CX-Register enthält vor dem Aufruf das Dateiattribut. Der Inhalt der Register BX, CX, SI, DI, BP, CS, SS, ES ändert sich durch den Funktionsaufruf nicht. Die Funktionen 5Bhex bis 61hex sind für MS-DOS reserviert.

**Funktion 62hex: Adresse des PSP ermitteln**

Die Funktion 62hex stellt die Adresse des aktuellen Programm Segment Präfix (PSP) fest. Nach dem Aufruf der Funktion gibt das BX-Register die Segmentadresse des PSP an. Um die Funktion 62hex aufzurufen, muß das AH-Register den Wert 62hex enthalten. Der Inhalt der Register AX, CX, DX, SI, DI, BP, CS, DS, SS und des Kennzeichenregisters ändert sich durch den Funktionsaufruf nicht.

Aufgrund der Aufwärtskompatibilität ist es sinnvoll, die veralteten CP/M-kompatiblen Funktionen durch Xenix-kompatible zu ersetzen. Eine Aufstellung der vergleichbaren Funktionen finden Sie in Tabelle 1.

Der Aufruf einer Betriebssystemroutine ist leider nicht immer von Erfolg gekrönt. Aus diesem Grund gibt der Interrupt 21hex eine Fehlermeldung aus. Die Fehlermeldungen handhaben die Xenix- und CP/M-kompatiblen Funktionen unterschiedlich. Während bei den CP/M-kompatiblen Funktionen das AL-Register durch den Wert FFhex das Auftreten eines Fehlers anzeigt, setzen die Xenix-kompatiblen Funktionen in diesem Fall das Carry-Bit des Kennzeichenregisters und geben den Fehlercode im AX-Register zurück. Tabelle 2 gibt die Bedeutungen der verschiedenen Fehlercodes an.

Im Zusammenhang mit der Funktion 59hex können Sie einen erweiterten Fehlercode abfragen. Tabelle 3 enthält zu dieser Funktion alle notwendigen Informationen. Das AX-Register (Tabelle 2) beschreibt den Fehler, das BH-Register die Fehlerursache und das CH-Register die Fehlerquelle. Mit dem BL-Register stellen Sie fest, welche Vorgehensweise Ihnen das Betriebssystem als Reaktion auf den Fehler vorschlägt.

Wie Sie gesehen haben, erhalten Sie mit dem MS-DOS-Interrupt 21hex eine komplette Betriebssystembibliothek, die alle wichtigen Funktionen beinhaltet, die für den Programmierer von Bedeutung sind. Aus diesem Grund verfügen auch fast alle modernen Programmiersprachen über Befehle, mit denen Sie den MS-DOS-Interrupt 21hex direkt aufrufen können, sowie zuvor einzelne Register setzen und nach dem Aufruf die Werte der gewünschten Register abfragen können.

(Markus Zietlow/ma)

Copro

Copro Computer

Individuelle AT-kompatible Komplettsysteme

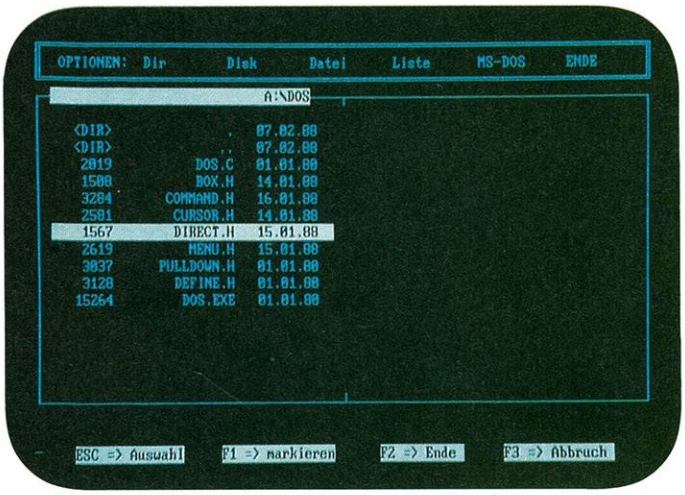
Wir zeigen Ihnen den Unterschied !
In Preis, Leistung und Service.

Natürlich liefern wir auch XT´s, Drucker und Zubehör. Unser Info gibt es kostenlos.
Schreiben Sie uns oder rufen Sie doch einfach an. Auch Händleranfragen sind uns stets willkommen.

**Copro GmbH • Berger Straße 15 • 6369 Niederdorfelden • Tel. 06101-3890**

Ihr Ansprechpartner: Herr Krieg

# Speicherresidente Programme – Zauberlehrlinge im Hintergrund




**Beispiel für ein Speicherresidentes Programm: Der DOS-Commander**

**Speicherresidente Programme wie Sidekick, PC Tools oder Superkey sind in aller Munde. Viele Programmierer vermeiden jedoch das Schreiben von speicherresidenter Software, da sie etwas Geheimnisvolles in sich birgt. Im folgenden erfahren Sie, wie diese Zauberlehrlinge funktionieren und in Assembler oder Turbo Pascal 4.0 zu programmieren sind.**

Das Wort Zauberlehrling ist in diesem Zusammenhang nicht ohne Grund gewählt. Zum einen lassen sich sehr schöne Hilfen mit diesen speicherresidenten Programmen herstellen, zum anderen entsteht bei unvorsichtiger Programmierung ein sehr instabiles System. Im folgenden wird die inzwischen gebräuchliche Bezeichnung TSR für speicherresidente Programme benutzt. TSR ist die Abkürzung des Interrupts von MS-DOS, der das Programm resident lädt: Terminate but Stay Resident (Beende das Programm, aber lasse es im Speicher).

Zunächst einmal sollten Sie sich mit der Speicherverwaltung auseinandersetzen, da die Kenntnisse darüber das Verständnis für die Abläufe im Computer und die hiermit verbundenen Probleme der TSR-Programmierung wesentlich erhöhen. Beispiele in Assembler und Turbo Pascal machen von diesen Kenntnissen dann ausgiebig Gebrauch.

## Die Entwicklung speicherresidenter Programme setzt Kenntnisse über die Speicherverwaltung voraus

Das Betriebssystem hat unter anderem die Aufgabe, den zur Verfügung stehenden Speicher zu verwalten. Hierfür stehen dem Anwender drei Funktionen zur Verfügung (Bild 1):

1) Funktion 48hex
– »Allocate memory«
Diese Funktion dient dem Reservieren eines Speicherplatzbereiches. Steht nicht genug Speicherplatz zur Verfügung, so liefert diese Funktion die maximal verfügbare Größe als Ergebnis zurück. Die Größenangaben erfolgen bei diesen Funktionen immer in Paragraphen. Ein Paragraph umfaßt 16 Byte. Somit ist die kleinste durchführbare Reservierung auf 16 Byte begrenzt.

2) Funktion 49hex
– »Free Allocated Memory«
Sie gibt den durch die Funktion 48hex reservierten Bereich an das Betriebssystem zurück.

3) Funktion 4Ahex – »Set Block«
Als Bezeichnung findet man auch »Modify Allocated Memory«. Die Größe des angegebenen Speicherbereiches ist veränderbar. Viele Programme nutzen diese Funktion um Raum zu schaffen für das Aufrufen eines Programms innerhalb der eigenen Umgebung (zum Beispiel der Shell-Befehl in Basic).

```
Allocate Memory
  Eingabe AH = 48H
          BX - Anzahl benötigter Paragraphen
  Ausgabe Carry-Bit gesetzt (Fehler)
          AX = 7 Memory Control Block defekt
          AX = 8 nicht genug Speicher vorhanden
          BX = Anzahl möglicher Paragraphen
          Carry-Bit gelöscht
          AX - Segment Adresse des reservierten
               Speicherblocks

Free Allocated Memory
  Eingabe AH = 49H
          ES - Segment Adresse des zurückzugebenden
               Speicherblocks
  Ausgabe Carry-Bit gesetzt (Fehler)
          AX = 7 Memory Control Block defekt
          AX = 9 falsche Segment Adresse
          Carry-Bit gelöscht
          keine Ausgabe

Set Block
  Eingabe AH = 4AH
          BX - Anzahl benötigter Paragraphen
          ES - Segment Adresse des zu ändernden
               Speicherblocks
  Ausgabe Carry-Bit gesetzt (Fehler)
          AX = 7 Memory control Block defekt
          AX = 8 Zu wenig Speicher
          AX = 9 falsche Segment Adresse
          BX - Anzahl möglicher Paragraphen
          Carry-Bit gelöscht
          keine Ausgabe

GET/SET Allocation Strategy
  Eingabe AH = 58H
          AL = 0 Get Strategy
               1 Set Strategy
          BX (AL=1) = 0\ First Fit
                      1 \Best Fit
                      2  Last Fit
  Ausgabe Carry-Bit gesetzt (Fehler)
          AX = 1  falscher Funktionscode
          Carry-Bit gelöscht
          AX (AL=0) = 0  First Fit
                      1  Best Fit
                      2  Last Fit


Program Segment Prefix

00H  INT 20 Programm Terminate Address
02H  Größe des resevierten Speichers
05H  Long Call zum DOS Interrupt (INT 21)
0AH  Terminate Address IP,CS
0EH  Control Break Adress IP,CS
12H  Critical error handler IP,CS
2CH  Environment Segment
36H  Code Segment
5CH  File Control Block I
6CH  File Control Block II
80H  Bei Aufruf Kommandozeile, Verwendung als
     Disk Transfer Area
```

**Bild 1. Funktionen, die das Betriebssystem für die Speicherverwaltung zur Verfügung stellt**

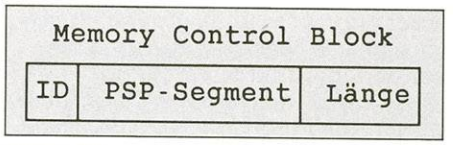



**Bild 2. Die Aufteilung eines Memory Control Blocks**

Die Funktionen lassen erkennen, daß das Betriebssystem irgendwie über den vorhandenen Speicher Buch führen muß. Dies geschieht durch eine einfache Liste sogenannter »Memory Control Blocks«.

Ein Memory Control Block (nachfolgend mit MCB bezeichnet) besteht aus 16 Byte. Das erste Byte dient zur Identifizierung, Byte zwei und drei enthalten die Segmentadresse des zugehörigen Programms und Byte vier und fünf die Länge des Speicherbereiches in Paragraphen (Bild 2).

## Speichervergabe nach Maß – der Programmierer bestimmt die Strategie

Die Strategie der Speichervergabe ist durch die Funktion 58hex sogar wählbar. Ist die Segmentadresse (Byte zwei und drei) gleich Null, so handelt es sich um einen nicht belegten Speicherbereich. Liegen durch die Rückgabe von Speicherbereichen mehrere solcher Bereiche direkt hintereinander, so faßt MS-DOS sie zu einem großen Bereich zusammen. Um nun die Aufteilung des Speichers zu sehen, benötigt man nur den Anfang der Liste. Diesen bekommt man zur Zeit durch die MS-DOS-Funktion 52hex. Zur Zeit deswegen, weil es sich hierbei um eine undokumentierte Funktion handelt, die sich in späteren MS-DOS-Versionen ändern könnte. Ausgehend vom ersten MCB erhält man durch Addition der jeweiligen Länge den nächsten Memory Control Block (Bild 3).

Diese Tatsachen nutzt das Programm »memory.pas« (Listing 1), um über die Speicherbelegung Auskunft zu geben. Für unsere Zwecke ist dies interessant, da Sie so überwachen können, ob sich Ihr Programm tatsächlich noch im Speicher befindet. Die Umgebung eines Programms besteht zusätzlich noch aus einem Environment-Bereich. In diesem Bereich sind die Pfade angegeben und die Variablen gespeichert, die durch den Set-Befehl vom MS-DOS aus deklariert wurden. Ab MS-DOS 3.0 wird hier auch der Name des Programms mit abgelegt, so daß wir versuchen, ihn zusätzlich anzuzeigen. Die Segmentadresse des Environments findet man auf Adresse 2Chex des PSP (»Program Segment Prefix«), auf das der MCB zeigt. Das PSP umfaßt 256 Byte und wird zu jedem Programm angelegt. Wenn Sie das Programm »Memory« laufen lassen, stellen Sie fest, daß sich schon einiges im Speicher befindet. In der Regel sind dies das MS-DOS selber, der Tastaturtreiber »keybgr.com« und der residente Teil von »command.com«.

## (K)ein Sprung ins kalte Wasser – Ihr erstes TSR-Programm

Versuchen Sie nun die ersten Schritte in Richtung TSR-Programmierung. Sie müssen dazu ein Programm schreiben und dieses dann speicherresident beenden. Da Sie aber nicht nur den Speicher langsam aber sicher zupflastern sollten, müssen Sie noch eine Aktivierung vorsehen. Hierfür eignet sich in der Kennenlernphase besonders gut der Interrupt 5 (»Print Screen«). Dieser Interrupt ist zu genau definierten Zeitpunkten auszulösen (Drücken der Taste [PrtScr]). Er ist zu keinem Zeitpunkt zeitkritisch, da er nur diese neue Aufgabe zu erledigen hat. Für das »Leben« des Systems ist er auch nicht wichtig, und man kann sicherlich bis zum nächsten Warm- oder Kaltstart auf eine Hardcopy verzichten.

Die Ausführungen in Assembler beschränken sich auf Com-Dateien, um die Sache nicht unnötig zu komplizieren. Dabei muß der grundsätzliche Aufbau einer Com-Datei so aussehen, wie dies Bild 4 zeigt.

Wichtig bei der Programmierung ist, daß man sich grundsätzlich nicht darauf verlassen darf, daß die Segmentregister bei der TSR-Programmausführung so stehen, wie Sie das gerne hätten. Daher ist eine Adressierung immer relativ zum CS (Codesegment) vorzunehmen, um auch wirklich die Variablen zu verändern, die dafür vorgesehen sind. Sonst besteht die Gefahr, daß der Computer sich sehr schnell »verläuft«.

Ein erstes Beispiel ist im Listing 2 »int05.asm« gegeben. Probieren Sie es aus. Geben Sie den Programmtext mit einem Textsystem ein. Anschließend starten Sie den Assembler:

```
masm int05,,,, [Enter]
```

dann gelinkt

```
link int05,,,, [Enter]
```

und in eine Com-Datei übersetzt

```
exe2bin int05.exe int05.com [Enter]
```

Da diese Schritte durch Tipp- oder Programmierfehler häufiger ausgeführt werden müssen, empfiehlt sich eine Batch-Datei, die durch folgende Befehle zu erzeugen ist:

```
copy con m.bat [Enter]
masm %1,,,, [Enter]
link %1,,,, [Enter]
exe2bin %1.exe %1.com [F6] [Enter]
```

»%1« steht für den Dateinamen und wird automatisch beim Aufruf durch den angegebenen Namen ersetzt. Geben Sie aber bitte nur den Namen ohne Erweiterung an:

```
m int5 [Enter]
```

und nicht

```
m int5.asm [Enter]
```

War alles erfolgreich, so dürfen Sie »int05« aufrufen.

Betrachten Sie anschließend die Speicherbelegung mit dem Programm »Memory«, und Sie stellen fest, daß unser Programm noch im Speicher ist. Die Funktion überprüfen Sie bitte durch Drücken der Taste [PrtScr].

So wie dies programmiert wurde, ist der Interrupt nur für diese spezielle Aufgabe zuständig. Da dies aber nicht immer so ist, nun zur nächsten Hürde. Hierfür benutzen Sie den Interrupt 16hex (Tastaturinterrupt). Um die verschiedenen Arten und Fähigkeiten aufzuzeigen, wurden zwar keine ausschließlich sinnvollen, dafür aber wirkungsvolle Programme gewählt.

Das erste Programm »int16.asm« (Listing 3) installiert Ihre Routine vor dem eigentlichen Tastaturtreiber und gibt nach 16 Buchstaben jeweils eine Meldung aus. Beim zweiten Programm »int16a.asm« (Listing 4) installieren Sie das Programm quasi zwischen den Interrupt. Abhängig vom Wert des AH-Registers wird die

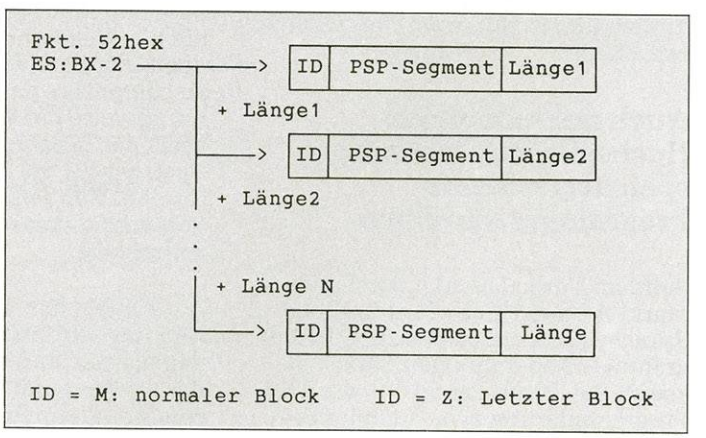



**Bild 3. Die MS-DOS-Funktion 52hex gibt Auskunft über die Aufteilung des Speichers**

```
CODE    Segment
; Zuordnung der Segmentregister zum Code-Segment
        ASSUME CS:CODE
        ASSUME DS:CODE
        ASSUME ES:CODE
        ASSUME SS:CODE
; Die ersten 100H Byte sind für das PSP reserviert
; (Bei COM-Dateien)
        ORG 100H
; Hier fängt das Programm an und verzweigt sofort
; zur Installationsroutine
START: JMP INSTALL
; Hier können Daten untergebracht werden
INTNR   DB  0           ; Nummer des zu benutzenden
                        ; Interrupts
OLDINT DD   0           ; alte Interruptadresse
                        ; zwischenspeichern

        Datenbereich

; und natürlich die TSR-Routine
NEUINT:  TSR-Routine

; Der Installationsteil ist wichtig!
; Er ist hier so ausgelegt, daß die Programme
; immer etwas mit Interrupt-Routinen zu tun haben
INSTALL:
; DOS-Funktion 35H: Get Interrupt Vektor
        MOV AH,35H
        MOV AL,INTNR
        INT 21H
; Zwischenspeichern des alten Interrupts
        MOV WORD PTR CS:[OLDINT],BX
        MOV WORD PTR CS:[OLDINT+2],ES
; DOS-Funktion 25H: Set Interrupt Vektor
        PUSH CS
        POP DS
        MOV DX,OFFSET NEUINT
        MOV AH,25H
        MOV AL,INTNR
        INT 21H
; Lade Addresse von der Install-Routine
; (= Länge des TSR-Programms)
        LEA DX,INSTALL
; DOS-Interrupt 27H (Terminate But Stay Resident)
        INT 27H
CODE    ENDS
        END  START
```

**Bild 4. So sieht der grundsätzliche Aufbau einer Com-Datei in Assembler aus**

originäre Interruptroutine sofort angesprungen oder aber als Unterprogramm aufgerufen. Dies erlaubt die Installation der »Hotkeys«: Das im Speicher befindliche Programm wird mit einer bestimmten Tastenkombination zum Leben erweckt.

Wenn Sie die Beispiele bis hierhin verfolgt haben, steht Ihnen vielleicht der Sinn nach etwas Sinnvollerem. Schauen Sie sich den Druckertreiber »olympia.asm« (Listing 5) an, der die Umlaute vom IBM-Zeichensatz auf Standard-ASCII umsetzt, um so Typenradmaschinen am Computer zu betreiben.

## Auch mit modernen Hochsprachen lassen sich speicherresidente Programme schreiben

Seitdem Turbo Pascal 4.0 auf dem Markt ist, steht eine volkstümliche Sprache zur Verfügung, die TSR-Programme ohne unnötigen Ballast erzeugt. Bei Turbo Pascal 3.0 war TSR-Programmierung zwar auch im Bereich des Möglichen (alles war irgendwie machbar ...), aber die Laufzeitbibliothek wurde immer mitgeschleppt, von dem Programmieraufwand einmal ganz abgesehen.

Die Version 4.0 kennt nun den Befehl »keep«, um Programme resident zu machen. Starten Sie solche Programme aber nie aus der Entwicklungsumgebung von Turbo Pascal, da sich das Programm und der Compiler (beziehungsweise die Entwicklungsumgebung) sonst gründlich ins Gehege kommen. Vergessen Sie auch nie den Heap und den Stack durch den Compilerbefehl »{$M 1024,0,0}« zu begrenzen, da MS-DOS beziehungsweise der Kommandoprozessor sonst den gesamten noch verfügbaren Speicher für das Pascal-Programm reserviert. Für den Rest und auch für »command.com« bleibt dann nichts mehr übrig.

Die Fähigkeiten von Turbo Pascal Routinen als Interrupt-Routinen zu deklarieren sind zwar schön, aber leider nur begrenzt zu gebrauchen. Beispiele sind die Programme »program1.pas« und »program2.pas«. »program1« (Listing 6) benutzt wieder den Interrupt 5, »program2« (Listing 7) den Interrupt 8 (»Clock Interrupt«). Die Programme entsprechen den Assembler-Programmen, sind aber vom Aufbau her einfacher zu lesen. Probleme treten – wie schon bei der Programmierung in Assembler – bei der Stack-Verwaltung auf. Der Compiler fügt bei einer Interrupt-Prozedur nur Code zum Sichern der Register ein, ändert aber nicht den Stack auf den Pascal-Stack. Dadurch kommt es sehr schnell zu einem Stacküberlauf, und damit »verabschiedet« sich das System.

## Die korrekte Stack-Verwaltung liegt in der Verantwortung des Programmierers

Hier muß wieder ein bißchen getrickst werden. Es ist wie im wirklichen Leben, die wichtigsten Dinge müssen Sie eben selber erledigen. Auch in Turbo Pascal 4.0 sollen Sie gleich etwas Sinnvolles auf die Beine stellen: Ein Programm, das beim Drücken einer bestimmten Taste (»Hotkey«) Datum und Uhrzeit anzeigt, bis Sie eine weitere Taste betätigen. Um den Bildschirmaufbau nicht zu zerstören, speichern Sie den alten Bildschirm in einer Variablen, und am Ende des Programms restaurieren Sie den Bildschirm durch Zurückschreiben des Inhaltes dieser Variablen wieder. Die in das Programm eingebauten Inline-Befehle dienen nur der korrekten Stack-Verwaltung. Schauen Sie sich das Listing 9 (»program3.pas«) an, und Sie erkennen vieles aus den Assemblerprogrammen wieder. Für eigene Anwendungen ist nur die Prozedur »neuprogramm« zu verändern. Um Speicherplatz zu sparen, lassen sich in einem Pascal-Programm auch mehrere Anwendungen realisieren, die jeweils über eine bestimmte Taste aktiviert werden.

## Beim Umgang mit TSR-Routinen sollte man stets Vorsicht walten lassen

Um Probleme zu vermeiden, verwenden Sie innerhalb der Prozedur »int16« einen kleinen Trick. Während der Ausführung Ihrer Routine setzen Sie den Interrupt-Vektor wieder auf die ursprüngliche Interrupt-Routine zurück. So überlisten Sie zum einen das MS-DOS, und zum zweiten ruft sich Ihre Routine nicht immer wieder selber auf.
Abschließend noch einige Bemerkungen zu TSR-Programmen. Grundsätzlich ist Vorsicht und Mißtrauen geboten. Durch das »Verbiegen der Inter-

# Ihr Partner für die professionelle Personalcomputertechnik

## Hardware

- Personalcomputer
- Bildschirme
- Drucker
- Druckerzubehör
- Grafik Ein-/Ausgabesysteme
- Festplatten
- Streamer
- Diskettenlaufwerke
- Co-/Ersatzprozessoren
- RAM-Chips
- Disketten
- Telekommunikation
- Erweiterungskarten

## Software

- Betriebssysteme
- GEM-/MS-Windows Software
- Utilities/Kommunikation
- Datenbanksysteme
- Datenverwaltung
- Integrierte Systeme
- Planungssoftware
- Statistik-/Mathematiksoftware
- Grafiksoftware/CAD
- Textverarbeitung
- Desktop Publishing
- Programmiersprachen
- Betriebswirtschaftliche Anwendungen

**PC PRODUCTS**

Personal Computer, Hardware, Software, Peripherie.
Maria Anna Hille, Bahnhofstraße 1, Postfach 1473,
D-7030 Böblingen, Tel. 07031/26013, Tlx. 7265745

**Fordern Sie bitte unsere kostenlose Gesamtpreisliste mit über 1000 Hard- und Softwareprodukten für den IBM PC, XT, AT, PS/2 und kompatible Personalcomputer an.**

---

☐ Senden Sie uns bitte Ihre kostenlose Gesamtpreisliste mit über 1000 Produkten für den IBM PC und Kompatible.

☐ Senden Sie uns bitte Unterlagen über folgende Produkte:

_____

_____

_____

**PC Products**
**Maria Anna Hille**
**Postfach 1473**

**D-7030 Böblingen**

Absender:

_____

_____

_____

_____

rupts« (inzwischen Fachjargon) besteht die Gefahr, daß sich verschiedene TSR-Routinen, die den gleichen Interrupt benutzen, ins Gehege kommen. Bekannt ist zum Beispiel, daß Sidekick sich nicht mit allen Programmen verträgt.

Wünschenswert ist auch die Fähigkeit, ein TSR-Programm wieder desaktivieren zu können, um solche Effekte zu vermeiden. Die Verzweigung zum originären Interrupt-Programm muß immer vorgenommen werden, es sei denn, Sie ersetzen dieses Programm vollständig. Beim Clock-Interrupt (int 8) ist dies für das System sogar »lebenswichtig«! Der Clock-Interrupt ist auch sehr zeitkritisch. Innerhalb der Bearbeitung der Ersatzroutine tritt eventuell schon wieder ein Interrupt auf, der das Programm dann sozusagen überrennt. Um Fehlfunktionen zu vermeiden, muß die Ersatzroutine erkennen, ob sie bereits aktiv ist. In diesem Fall muß dann die Original-Routine direkt den Aufruf übernehmen.

Besondere Aufmerksamkeit verdient der Stack. Da hiervon im allgemeinen immer zu wenig vorhanden ist, müssen Sie unbedingt einen eigenen Stack vorsehen. Innerhalb der Interrupt-Routinen sollten Sie MS-DOS-Aufrufe sehr vorsichtig handhaben, da MS-DOS bei Mehrfachaufrufen von Funktionen nicht gewährleistet, daß die Registerinhalte unverändert bleiben.

Werden MS-DOS-Funktionen während ihrer Ausführung durch einen Interrupt unterbrochen und erneut aufgerufen, dann kann es durch undefinierte Registerinhalte zu nichtreproduzierbaren Ergebnissen kommen.

(Helmut Amshove/zi)

```
[ZM5]    0: program memory;
[706]    1: (* ---------------------------------- *)
[CLC]    2: (* Demonstration der Speicherbelegung *)
[706]    3: (* ---------------------------------- *)
[C96]    4: (* H.Amshove                  6/88   *)
[706]    5: (* ---------------------------------- *)
[000]    6:
[UW3]    7: uses crt,dos;
[000]    8:
[XH6]    9: type MCB_Ptr = ^MCB_Typ;
[DD5]   10:      MCB_Typ = record
[RN2]   11:                  Id:char;
[VF5]   12:                  PSP_Segment:word;
[RA3]   13:                  Laenge:word;
[TU0]   14:                end;
[000]   15:
[1K7]   16: var Reg           : registers;
[AS4]   17:     MCB           : MCB_Ptr;
[6N6]   18:     Curent_MCB    : MCB_Ptr;
[ED3]   19:     Ch            : char;
[WT5]   20:     EnvSeg,EnvOfs : word;
[4M6]   21:     FreierSpeicher: word;
[000]   22:
[706]   23: (* ---------------------------------- *)
[000]   24:
[FZ7]   25: function FirstMCB:MCB_Ptr;
[BH1]   26: begin
[A13]   27:  Reg.AX:=$5200;
[K82]   28:  MSDOS(REG);
[43B]   29:  FirstMCB:=ptr(Memw[reg.ES:reg.bx-2],0);
[TU0]   30: end;
[000]   31:
[706]   32: (* ---------------------------------- *)
[000]   33:
[A4B]   34: function DezToHex(j:word):string;
[YSD]   35: const HexDigit:array[0..15] of char = '0123
           456789ABCDEF';
[000]   36:
[BH1]   37: begin
[27H]   38: DezToHex:= HexDigit[j shr 12]            +
           HexDigit[(j shr 8) and $0f]+
[G4F]   39:            HexDigit[(j shr 4) and $000f] +
           HexDigit[j and $000f];
[TU0]   40: end;
[000]   41:
[706]   42: (* ---------------------------------- *)
[CS7]   43: (*  Hauptprogramm                     *)
[706]   44: (* ---------------------------------- *)
[000]   45:
[BH1]   46: begin
[NF2]   47:  ClrScr;
[FH3]   48:  MCB:=FirstMCB;
[CX9]   49:  writeln('MCB      ID   PSP-     Länge   En
           v ');
[Y5F]   50:  writeln('Segment       Segment          Se
           gment     Name');
[D3D]   51:  writeln('-----------------------------------
           ------------------------------');
[X52]   52:  repeat
[LF7]   53:     write(DezToHex(seg(mcb^)));
[VHP]   54:     write('     ',mcb^.id,'     ',DezToHex(m
           cb^.psp_segment),'     ',DezToHex(mcb^.laeng
           e));
[37A]   55:     EnvSeg:=MemW[MCB^.psp_segment:$2c];
[BN2]   56:     EnvOfs:=0;
[FP4]   57:     if EnvSeg<>0 then
[BH1]   58:      begin
[NU4]   59:       write('    ',EnvSeg);
[X52]   60:       repeat
[WY4]   61:        EnvOfs:=EnvOfs+1;
[EV8]   62:       until MemW[EnvSeg:EnvOfs]=$0000;
[WY4]   63:       EnvOfs:=EnvOfs+2;
[KF9]   64:       if MemW[EnvSeg:EnvOfs]=$0001 then
[BH1]   65:         begin
[WY4]   66:          EnvOfs:=EnvOfs+2;
[5V2]   67:          write(' ':5);
[X52]   68:          repeat
[S59]   69:           write(chr(mem[EnvSeg:EnvOfs]));
[WY4]   70:           EnvOfs:=EnvOfs+1;
[KN8]   71:          until (mem[EnvSeg:EnvOfs]=$00);
[TU0]   72:        end;
[TU0]   73:      end;
[9Z2]   74:    writeln;
[Z0B]   75:    MCB:=ptr(seg(MCB^)+succ(mcb^.laenge),0);
[XR4]   76:  until (MCB^.id='Z');
[T3W]   77:  writeln(DezToHex(seg(mcb^)),'  ',mcb^.id,'
           ',DezToHex(mcb^.psp_segment),' ',DezToHex(
           mcb^.laenge));
[5T0]   78: end.
[000]   79:
```

**Listing 1. Ein Programm zur Demonstration der Speicherbelegung**

```
[4R8]    0: ;-----------------------------------------
           -----------;
[XP8]    1: ;     E I N F A C H E   T S R - R O U T I N
           E          ;
[WL8]    2: ; ----------------------------------------
           ---------- ;
[R78]    3: ;  H.Amshove                              6/
           8          ;
[4R8]    4: ;-----------------------------------------
           -----------;
[000]    5:
[24C]    6: CODE   SEGMENT
           ; Segmentzuordnung
[H73]    7:        ASSUME CS:CODE
[H73]    8:        ASSUME DS:CODE
[H73]    9:        ASSUME ES:CODE
[H73]   10:        ASSUME SS:CODE
[DD1]   11:        ORG 100H
[KUG]   12: START: JMP INSTALL
           ; bei Aufruf nur installieren
[000]   13:
[K2A]   14: INTNR   DB  05H
           ; Interrupt Nr
[KYB]   15: OLDINT  DD  0
           ; alte Interrupt Nr
[A7C]   16: LANG    DB  30
           ; Laenge der Meldung
[D1F]   17: OUTSTR  DB  'Dies ist die erste TSR-Routine
           ' ; Meldung
[TH7]   18: ;-----------------------------------------
           ------;
[2C7]   19: ;   N e u e r   I N T 5
                ;
```

```
[TH7]   20: ;-----------------------------------------
            ------;
[000]   21:
[RS3]   22: NEUINT:  PUSH CX
[AU1]   23:          PUSH DI
[1X1]   24:          PUSH AX
[KC9]   25:          MOV CH,0
          ; Zähler für LOOP
[MKA]   26:          MOV CL,CS:LANG
          ; initialisieren
[LH8]   27:          MOV DI,0
          ; Offset auf 0
[FDA]   28:          MOV AH,14
          ; Bios Fkt 14 von INT 10
[W9C]   29: AUS:     MOV AL,CS:OUTSTR[DI]
          ; Zeichen laden
[RT7]   30:          INT 10h
          ; ausgeben
[B0A]   31:          INC DI
          ; nächstes Zeichen
[GW1]   32:          LOOP AUS
[LD1]   33:          POP AX
[8B1]   34:          POP DI
[XD1]   35:          POP CX
[MS8]   36:          IRET
          ; Interrupt Ende
[H0A]   37: ;-----------------------------------------
            ----------------------;
[YVD]   38: ; Installationsteil   TSR-Routine
                                ;
[H0A]   39: ;-----------------------------------------
            ----------------------;
[XSE]   40: INSTALL:    MOV  AH,35H
          ; Lesen alten Interrupt
[UU2]   41:             MOV  AL,INTNR
[4C1]   42:             INT  21H
[YTF]   43:             MOV  WORD PTR CS:[OLDINT]  , BX
          ; Speichern alten Interrupt
[5A7]   44:             MOV  WORD PTR CS:[OLDINT+2], ES
[AW1]   45:             PUSH CS
[VP1]   46:             POP  DS
[1CD]   47:             MOV  DX,OFFSET NEUINT        ;
          Setzen neuer Interrupt
[ZZ1]   48:             MOV  AH,25H
[UU2]   49:             MOV  AL,INTNR
[4C1]   50:             INT  21H
[KHB]   51:             LEA  DX,INSTALL              ;
          Länge des Programms
[LVC]   52:             INT  27H                     ;
          Beenden und resident halten
[EN2]   53:   CODE      ENDS
[ZY1]   54:             END START
[000]   55:
[000]   56:
```

**Listing 2. Ein TSR-Programm, das durch die PrtScr-Taste aktiviert wird**

```
[4R8]    0: ;-----------------------------------------
            ----------;
[XP8]    1: ;     E I N F A C H E   T S R - R O U T I N
            E          ;
[WU8]    2: ;              INT 16  1.Versuch
                        ;
[R2G]    3: ;     Installation vor dem eigentlichen Int
            errupt     ;
[WL8]    4: ; ----------------------------------------
            ---------- ;
[R78]    5: ;  H.Amshove                              6/8
            8          ;
[4R8]    6: ;-----------------------------------------
            ----------;
[000]    7:
[24C]    8: CODE   SEGMENT
            ; Segmentzuordnung
[H73]    9:        ASSUME CS:CODE
[H73]   10:        ASSUME DS:CODE
[H73]   11:        ASSUME ES:CODE
[H73]   12:        ASSUME SS:CODE
[DD1]   13:        ORG 100H
[KUG]   14: START: JMP INSTALL
            ; bei Aufruf nur installieren
[000]   15:
[KX9]   16: INTNR    DB  16H
            ; Interrupt Nr
[T5B]   17: OLDINT   DD  0
            ; alter Interrupt
```

```
[EMC]   18: OLDSS    DW  0
            ; alten Stack sichern
[WG2]   19: OLDSP    DW  0
[KXC]   20: LAENGE   DB  0AH
            ; Laenge des Strings
[C3A]   21: MELDUNG  DB  'DOS ist OK'
            ; Meldung
[65E]   22: ZAEHLER  DW  0
            ; Zähler für Tastendruck
[000]   23:
[TH7]   24: ;-----------------------------------------
            ------;
[WB7]   25: ;   N e u e r   I N T 1 6
                ;
[TH7]   26: ;-----------------------------------------
            ------;
[000]   27:
[000]   28:
[RPE]   29: NEUINT:    CMP AH,0                      ;
            Soll Taste gelesen werden
[EYB]   30:            JNE ENDE                      ;
            Nein, direkt zum Original
[BR9]   31:            CLI                           ;
            Sperre Interrupt
[G1A]   32:            MOV CS:OLDSS,SS               ;
            Stack sichern
[FU3]   33:            MOV CS:OLDSP,SP
[X83]   34:            INC CS:ZAEHLER
[H3B]   35:            CMP CS:ZAEHLER,10h            ;
            Zähler zu klein
[NMA]   36:            JL  ORIGINAL                  ;
            ab zum Interrupt
[U6C]   37:            PUSH CS                       ;
            Programm Stack setzen
[NT1]   38:            POP  SS
[872]   39:            MOV SP,100h
[U6A]   40:            PUSH AX                       ;
            Register sichern
[DX1]   41:            PUSH CX
[AU1]   42:            PUSH DI
[YZ9]   43:            MOV  CH,0                     ;
            Meldung ausgeben
[5M3]   44:            MOV CL,CS:LAENGE
[BN1]   45:            MOV DI,0
[AP1]   46:            MOV AH,14
[H27]   47:  AUS:      MOV AL,CS:MELDUNG[DI]
[EU1]   48:            INT 10h
[S91]   49:            INC DI
[GW1]   50:            LOOP AUS
[B1A]   51:            POP DI                        ;
            Register zurück
[XD1]   52:            POP CX
[LD1]   53:            POP AX
[GM3]   54:            MOV CS:ZAEHLER,0
[A83]   55: ORIGINAL:  CLI
[YAB]   56:            MOV SS,CS:OLDSS               ;
            alter Stack zurück
[UV3]   57:            MOV SP,CS:OLDSP
[PE0]   58:            STI
[CCF]   59: ENDE:      JMP CS:[OLDINT]               ; n
            un der normale Interrupt
[H0A]   60: ;-----------------------------------------
            ----------------------;
[YVD]   61: ; Installationsteil   TSR-Routine
                                ;
[H0A]   62: ;-----------------------------------------
            ----------------------;
[N7F]   63: INSTALL:    MOV AH,35H                    ;
            lesen alten Interrupt
[PE2]   64:             MOV AL,INTNR
[TL1]   65:             INT 21H
[LUF]   66:             MOV  WORD PTR CS:[OLDINT],BX  ;
            speichern alten Interrupt
[D07]   67:             MOV  WORD PTR CS:[OLDINT+2],ES
[AW1]   68:             PUSH CS
[4D1]   69:             POP DS
[EUF]   70:             MOV DX, OFFSET NEUINT         ; S
            etzen Adresse der neuen Routine
[VU1]   71:             MOV AH,25H
[MV1]   72:             MOV AL,16H
[TL1]   73:             INT 21H
[KHB]   74:             LEA  DX,INSTALL               ;
            Länge des Programms
[LVC]   75:             INT  27H                      ;
            Beenden und resident halten
[EN2]   76:   CODE      ENDS
[ZY1]   77:             END START
[000]   78:
[000]   79:
```

**Listing 3. Dieses Programm installiert sich vor den Tastaturtreiber**

```
[4R8]    0: ;-----------------------------------------
           -----------;
[XP8]    1: ;     E I N F A C H E  T S R - R O U T I N
           E         ;
[GY8]    2: ;          I N T  16    2.Versuch
                     ;
[2XD]    3: ;     Installation "zwischen" dem Interrupt
                     ;
[WL8]    4: ; ---------------------------------------
           ---------- ;
[R78]    5: ;  H.Amshove                              6/8
           8         ;
[4R8]    6: ;-----------------------------------------
           -----------;
[000]    7:
[24C]    8: CODE    SEGMENT
           ; Segmentzuordnung
[H73]    9:         ASSUME CS:CODE
[H73]   10:         ASSUME DS:CODE
[H73]   11:         ASSUME ES:CODE
[H73]   12:         ASSUME SS:CODE
[DD1]   13:         ORG 100H
[KUG]   14: START: JMP INSTALL
           ; bei Aufruf nur installieren
[000]   15:
[1W2]   16: INTNR     DB  16H
[T5B]   17: OLDINT    DD  0
           ; alter Interrupt
[BH2]   18: OLDSS     DW  0
[WG2]   19: OLDSP     DW  0
[8Y2]   20: LAENGE    DB  12h
[Y19]   21: Meldung   DB  'Taste F1 betätigt.'
[TH7]   22: ;-----------------------------------------
           ------;
[6B7]   23: ;   N e u e r   I N T 16
                 ;
[TH7]   24: ;-----------------------------------------
           ------;
[000]   25:
[000]   26:
[P7E]   27: NEUINT:     CMP AH,0
           ; auf Tastendruck warten
[7Z0]   28:             JE OK
[5CF]   29:             JMP CS:[OLDINT]
           ; nein also original Interrupt
[03G]   30: OK:         PUSHF
           ; ja also original Interrupt als
[RBB]   31:             CALL CS:[OLDINT]
           ; Unterprogramm
[82A]   32:             CMP AH,3BH
           ; unser "Hotkey"
[TB9]   33:             JNE ENDE
           ; nein, dann ende
[3YB]   34:             CLI
           ; ab hier die auszuführende
[GN8]   35:             MOV CS:OLDSS,SS
           ; Funktion
[FU3]   36:             MOV CS:OLDSP,SP
[AW1]   37:             PUSH CS
[NT1]   38:             POP  SS
[842]   39:             MOV SP,100H
[1X1]   40:             PUSH AX
[DX1]   41:             PUSH CX
[AU1]   42:             PUSH DI
[7P1]   43:             MOV  CH,0
[5M3]   44:             MOV CL,CS:LAENGE
[BN1]   45:             MOV DI,0
[AP1]   46:             MOV AH,14
[H27]   47:  AUS:       MOV AL,CS:MELDUNG[DI]
[EL1]   48:             INT 10H
[S91]   49:             INC DI
[GW1]   50:             LOOP AUS
[8B1]   51:             POP DI
[XD1]   52:             POP CX
[LD1]   53:             POP AX
[PD0]   54:             CLI
[1X3]   55:             MOV SS,CS:OLDSS
[UV3]   56:             MOV SP,CS:OLDSP
[PE0]   57:             STI
[B73]   58: ENDE:       IRET
[H0A]   59: ;-----------------------------------------
           ----------------------;
[YVD]   60: ; Installationsteil   TSR-Routine
                                 ;
[H0A]   61: ;-----------------------------------------
           ----------------------;
[N7F]   62: INSTALL:    MOV AH,35H                   ;
           lesen alten Interrupt
[PE2]   63:             MOV AL,INTNR
[TL1]   64:             INT 21H
[LUF]   65:             MOV  WORD PTR CS:[OLDINT],BX  ;
           speichern alten Interrupt
[D07]   66:             MOV  WORD PTR CS:[OLDINT+2],ES
[AW1]   67:             PUSH CS
[4D1]   68:             POP DS
[EUF]   69:             MOV DX, OFFSET NEUINT        ; S
           etzen Adresse der neuen Routine
[VU1]   70:             MOV AH,25H
[MV1]   71:             MOV AL,16H
[TL1]   72:             INT 21H
[KHB]   73:             LEA  DX,INSTALL               ;
           Länge des Programms
[LVC]   74:             INT  27H                      ;
           Beenden und resident halten
[EN2]   75:   CODE      ENDS
[ZY1]   76:             END START
[000]   77:
```

**Listing 4. Per Hotkey wird das Programm zum Leben erweckt**

```
[TH7]    0: ;-----------------------------------------
           ------;
[GN7]    1: ;      PRINTER ANPASSUNG
                  ;
[TH7]    2: ;-----------------------------------------
           ------;
[9B7]    3: : H.Amshove                               6/88
                  ;
[TH7]    4: ;-----------------------------------------
           ------;
[000]    5:
[WEC]    6: CODE    SEGMENT
              ; Segmentzuordnung
[H73]    7:         ASSUME CS:CODE
[H73]    8:         ASSUME DS:CODE
[H73]    9:         ASSUME ES:CODE
[H73]   10:         ASSUME SS:CODE
[DD1]   11:         ORG 100H
[CCE]   12: START: JMP INSTALL
              ; sofort installieren
[TH7]   13: ;-----------------------------------------
           ------;
[NFF]   14: UMLAUTE DB  81H,9AH,84H   ;ue,UE,ae
              ; Tabelle der Umlaute
[4BC]   15:         DB  8EH,94H,99H   ;AE,oe,OE
              ; und Ersetzungen
[9H3]   16:         DB  0E1H          ;ss
[793]   17:         DB  7DH,5DH,7BH
[NH1]   18:         DB  5BH
[H93]   19:         DB  7CH,5CH,7EH
[Y13]   20: UML     DB  3FH,00H
[ARC]   21: INTNR   DB  17H
              ; Druckerinterrupt
[HTF]   22: I17PTR  DD  0
              ; Zeiger auf original Interrupt
[T6D]   23: OLDSS   DW  0
              ; sichern Stacksegment
[H72]   24: OLDSP   DW  0
[HE7]   25: ;-----------------------------------------
           -----;
[V8B]   26: ; NEUER EINTRITTSPUNKT FUER BIOS INTERRUPT
           17    ;
[LF8]   27: ;   AH = 0   ZEICHEN IN AL DRUCKEN
                ;
[DU8]   28: ;   AH = 1   PRINTER INITIALISIEREN
                ;
[GS8]   29: ;   AH = 2   PRINTER STATUSABFRAGE
                ;
[2W8]   30: ;   DX=0,1,2  AKTUELLER PRINTER
                ;
[HE7]   31: ;-----------------------------------------
           -----;
[GR3]   32: NEUINT:  OR   AH,AH
[D3E]   33:          JNZ  BIOSJMP
              ; kein zu druckendes Zeichen
[DF8]   34:          TEST AL,80H
              ; Zeichen < 127
[R82]   35:          JZ   BIOSJMP
[01A]   36:          MOV  CS:OLDSS,SS
              ; Stack sichern
[0P3]   37:          MOV  CS:OLDSP,SP
[AW1]   38:          PUSH CS
[WK8]   39:          POP  SS
              ; neuer Stack
[GB2]   40:          MOV  SP,100H
[GW1]   41:          PUSH DS
```

```
[PW1]   42:           PUSH ES
[AU1]   43:           PUSH DI
[DX1]   44:           PUSH CX
[7X1]   45:           PUSH BX
[HE7]   46: ;-----------------------------------------
          -----;
[PD7]   47: ;   TEST AUF UMLAUT
                ;
[HE7]   48: ;-----------------------------------------
          -----;
[AW1]   49:           PUSH CS
[1R1]   50:           POP  ES
[9T3]   51:           LEA  DI,CS:UMLAUTE
[YT1]   52:           MOV  CX,7
[7D0]   53:           CLD
[F82]   54:           REPNE SCASB
[2A2]   55:           JNE  BIOSREG
[AK3]   56:           MOV  CL,CS:[DI+6]
[Y33]   57:           MOV  CS:UML,CL
[V63]   58:           MOV  AL,CS:UML
[PC7]   59: ;-----------------------------------------
          ----;
[LU8]   60: ;   Ersatz ausgeben
              ;
[PC7]   61: ;-----------------------------------------
          ----;
[VN1]   62:           MOV  AH,0
[HK3]   63: BIOSREG:  POP  BX
[SR1]   64:           POP  CX
[NM1]   65:           POP  DI
[1R1]   66:           POP  ES
[VP1]   67:           POP  DS
[GV3]   68:           MOV  SS,CS:OLDSS
[VU3]   69:           MOV  SP,CS:OLDSP
[SN5]   70: BIOSJMP:  JMP  CS:[I17PTR]
[XN8]   71: ;-----------------------------------------
          ------------;
[F8A]   72: ;   ENDE RESIDENTER PROGRAMMTEIL
                    ;
[XN8]   73: ;-----------------------------------------
          ------------;
[174]   74: INSTALL:  MOV AH,35H
[YT3]   75:           MOV AL,CS:INTNR
[TL1]   76:           INT 21H
[LZ6]   77:           MOV WORD PTR CS:[I17PTR  ],BX
[SZ6]   78:           MOV WORD PTR CS:[I17PTR+2],ES
[AW1]   79:           PUSH CS
[VP1]   80:           POP  DS
[NP4]   81:           MOV DX, OFFSET NEUINT
[VU1]   82:           MOV AH,25H
[YT3]   83:           MOV AL,CS:INTNR
[TL1]   84:           INT 21H
[973]   85:           LEA DX,INSTALL
[ED1]   86:           INT  27H
[4A2]   87: CODE      ENDS
[ZY1]   88:           END START
```

**Listing 5. Mit diesem Treiber lassen sich Typenraddrucker am Computer betreiben**

```
[C94]    0: program eins;
[D85]    1: (* ------------------------------ *)
[X78]    2: (*  TSR Programm über INT5        *)
[D85]    3: (* ------------------------------ *)
[W16]    4: (* H.Amshove                6/88  *)
[D85]    5: (* ------------------------------ *)
[000]    6:
[000]    7:
[ME2]    8: (*$M 1024,0,0 *)
[000]    9:
[W3C]   10: uses crt,DOS;          (* für KEEP, REGISTER
          S ,GETINT.. *)
[000]   11:
[WZ7]   12: var zaehler,anzahl     : word;
[RR8]   13:     cursorx,cursory    : word;
[000]   14:
[000]   15:
[D85]   16: (* ------------------------------ *)
[000]   17:
[5D9]   18: procedure Meldung;interrupt;
[PPM]   19: (* ersetzt Interrupt 5 und wird erst nach e
          iner bestimmten Anzahl
[996]   20:    Aufrufe aktiviert *)
```

```
[000]   21:
[BH1]   22: begin
[WX5]   23:  Zaehler:=Zaehler+1;
[717]   24:  if Zaehler=20 then begin
[5T5]   25:                        CursorX:=Wherex;
[LT5]   26:                        CursorY:=Wherey;
[W15]   27:                        Anzahl:=Anzahl+1;
[518]   28:                        gotoxy(1,1);write(anza
          hl);
[H73]   29:                        zaehler:=0;
[CZ7]   30:                        gotoxy(cursorx,cursory
          );
[TU0]   31:                      end;
[TU0]   32: end;
[000]   33:
[D85]   34: (* ------------------------------ *)
[000]   35:
[BH1]   36: begin
[H63]   37:  Zaehler:=0;
[VR2]   38:  anzahl:=0;
[B86]   39:  SetIntVec(5,@Meldung);
[BS1]   40:  Keep(0);
[5T0]   41: end.
```

**Listing 6. Dieses Pascal-Programm belegt den Interrupt 5**

```
[U84]    0: program zwei;
[D85]    1: (* ------------------------------ *)
[388]    2: (*  TSR Programm über INT8        *)
[D85]    3: (* ------------------------------ *)
[W16]    4: (* H.Amshove                6/88  *)
[D85]    5: (* ------------------------------ *)
[000]    6:
[000]    7:
[ME2]    8: (*$M 1024,0,0 *)
[000]    9:
[W3C]   10: uses crt,DOS;          (* für KEEP, REGISTER
          S ,GETINT.. *)
[000]   11:
[WZ7]   12: var zaehler,anzahl     : word;
[RR8]   13:     cursorx,cursory    : word;
[FW6]   14:     saveInt            : pointer;
[000]   15:
[000]   16:
[D85]   17: (* ------------------------------ *)
[000]   18:
[5D9]   19: procedure Meldung;interrupt;
[VHD]   20: (* wird in den Interrupt 8 mit eingebunden
          *)
[BH1]   21: begin
[0SG]   22:  inline($9c/$ff/$1e/saveInt);     (* Aufruf
          alten Interrupt *)
[WX5]   23:  Zaehler:=Zaehler+1;
[717]   24:  if Zaehler=20 then begin
[5T5]   25:                        CursorX:=Wherex;
[LT5]   26:                        CursorY:=Wherey;
[W15]   27:                        Anzahl:=Anzahl+1;
[5Y7]   28:                        gotoxy(1,1);write(Anza
          hl);
[H63]   29:                        Zaehler:=0;
[CZ7]   30:                        gotoxy(cursorx,cursory
          );
[TU0]   31:                      end;
[TU0]   32: end;
[000]   33:
[D85]   34: (* ------------------------------ *)
[000]   35:
[BH1]   36: begin
[H63]   37:  Zaehler:=0;
[VR2]   38:  anzahl:=0;
[Z76]   39:  GetIntVec(8,saveint);
[BV6]   40:  SetIntVec(0,saveint00);
[VK5]   41:  checkbreak:=false;
[NW6]   42:  setIntVec($1B,saveInt1B);
[M86]   43:  SetIntVec(8,@Meldung);
[BS1]   44:  Keep(0);
[5T0]   45: end.
```

**Listing 7. Diese Routine wird in den Clock-Interrupt eingebunden**

```
[W64]    0: program TSR16;
[TA6]    1: (* ------------------------------------ *)
[UEA]    2: (*   TSR Programm in Turbo-Pascal      *)
[UEB]    3: (*   hier: Verwendung des Interrupt 16  *)
[TA6]    4: (* ------------------------------------ *)
[CS6]    5: (*   H.Amshove                 6/88     *)
[TA6]    6: (* ------------------------------------ *)
[000]    7:
[SGE]    8: (*$M 1024,0,0 *)          (* Festlegung Sta
           ckgröße und Heap  *)
[000]    9:
[9AE]   10: uses crt,dos;             (* für GOTOXY,Set
           IntVec ....       *)
[000]   11:
[2DH]   12: const HotKey = $4400;     (* Dieser ScanCod
           e lößt die Funktion aus *)
[000]   13:
[YB9]   14: type Schirm=array[1..$4000] of byte;
[000]   15:
[THH]   16: var AltSP, AltSS       :   word;      (* Spe
           ichern des alten Stacks *)
[FLF]   17:     TurboSP,TurboSS    :   word;      (* Spe
           ichern Turbo-Stack      *)
[WSG]   18:     Intprog,                          (* Add
           resse der Interrupt Procedur *)
[YVG]   19:     AltInt16,                         (* Add
           resse des Original Interrupts *)
[GRG]   20:     NeuInt16           :   Pointer;   (* Add
           resse neuer Interrupt        *)
[NXK]   21:     Bildschirm1        :   Schirm absolute
           $B800:0;   (* Bildschirmspeicher *)
[FTF]   22:     Bildschirm2        :   Schirm;
                      (* Zwischenspeicher   *)
[DA4]   23:     Stunde,Minute,
[Z2F]   24:     Sekunde,S100       :   word;      (* Für
           Ermittlung Uhrzeit *)
[ET4]   25:     Tag,Monat,Jahr,
[PNE]   26:     Wochentag          :   word;      (* Für
           Ermittlung Datum   *)
[H0L]   27:     CursorX,CursorY    :   byte;      (* Spe
           ichern alte Cursor Position *)
[000]   28:
[000]   29:
[000]   30:
[XZN]   31: (* Diese Procedure wird in den Interrupt ei
           ngebunden und ruft unsere
[7HB]   32:    neue Procedur INTPROG auf
                                    *)
[000]   33:
[5PN]   34: procedure INT16(Flags,cs,ip,ax,bx,cx,dx,si,
           di,ds,es,bp:word);interrupt;
[000]   35:
[B2H]   36:   (* als Funktion ersparen wir uns die Über
           gabe von AX *)
[P0A]   37:   function AlterInterrupt:word;
[WZ8]   38:   inline($31/$c0/$9c/$FF/$1E/altint16);
[000]   39:
[BH1]   40: begin
[X0H]   41:  if hi(ax)=0 then                    (* Soll
            Taste eingegeben werden ? *)
[X52]   42:   repeat
[B7K]   43:     ax:=AlterInterrupt;              (* Führ
           e Tastaturinterrupt aus *)
[FWD]   44:     if ax=HotKey                     (* gedr
           ückte Taste = HotKey ? *)
[ZS2]   45:       then begin
[8XA]   46:                (* Turbo Stackbereich einschal
           ten *)
[VAB]   47:                inline($8c/$16/altSS/
                (* MOV [AltSS],SS  *)
[BG9]   48:                         $89/$26/altsp/
                (* MOV [AltSP],SP  *)
[G45]   49:                         $FA/
                (* CLI *)
[HXA]   50:                         $8E/$16/TurboSS/
                (* MOV SS,[TurboSS] *)
[4WA]   51:                         $8B/$26/TurboSP/
                (* MOV SP,[TurboSP] *)
[U95]   52:                         $FB);
                (* STI *)
[8YF]   53:                GetIntVec($16,NeuInt16);    (*
            kurz alles wieder beim alten *)
[3LB]   54:                SetIntVec($16,AltInt16);    (*
            lassen ...                   *)
[3D7]   55:                (*    Rufe unser Programm  *)
[Y9C]   56:                inline($ff/$1E/Intprog);
           (* CALL FAR [INTPROG] *)
[D1G]   57:                SetIntVec($16,NeuInt16);    (*
            jetzt wieder unsere Routine *)
[LU8]   58:                (* Stack  zurueckspeichern *)
[K67]   59:                inline($FA/
                (* CLI *)
[4L9]   60:                         $8E/$16/AltSS/
                (* MOV SS,[AltSS] *)
[BH9]   61:                         $8B/$26/AltSP/
                (* MOV SP,[AltSP] *)
[U95]   62:                         $FB);
                (* STI *)
[TU0]   63:              end;
[9Z4]   64:   until ax<>HotKey
[XL6]   65:   else (* Ende Interrupt *)
[RCD]   66:        inline ($a1/altint16+2/
                (* MOV AX,altint16+2 *)
[878]   67:                $87/$46/$10/
                (* XCHG AX,BP+10     *)
[VXA]   68:                $8B/$1E/altint16/
                (* MOV AX,altint16   *)
[HE8]   69:                $87/$5E/$0E/
                (* XCHG AX,BP+0E     *)
[A67]   70:                $5D/
                (* POP BP            *)
[E57]   71:                $07/
                (* POP ES            *)
[U67]   72:                $1F/
                (* POP DS            *)
[U57]   73:                $5F/
                (* POP DI            *)
[477]   74:                $5E/
                (* POP SI            *)
[7A7]   75:                $5a/
                (* POP DX            *)
[C67]   76:                $59/
                (* POP CX            *)
[747]   77:                $CB);
                (* RETF              *)
[TU0]   78: end;
[000]   79:
[000]   80:
[X51]   81: (*$F+ *)
[5RK]   82: (* muß als FAR deklariert werden, da wir mi
           t CALL FAR arbeiten *)
[0M7]   83: procedure Neuprogramm;
[000]   84:
[BH1]   85: begin
[5T5]   86:  CursorX:=Wherex;
[LT5]   87:  CursorY:=Wherey;
[5YK]   88:  Bildschirm2:=Bildschirm1;
              (* sichern alten Bildschirm *)
[NM2]   89:  clrscr;
[UFH]   90:  writeln('I n t e r u p t  16  : Sie haben
           die Taste F10 gedrückt ');
[X52]   91:  repeat
[TWA]   92:   gettime(Stunde,Minute,Sekunde,s100);
[1GA]   93:   getdate(Jahr,Monat,Tag,Wochentag);
[PUE]   94:   gotoxy(5,5); write('Datum:   ',Tag:2,'.',
           Monat:2,'.',jahr:4);
[F4K]   95:   gotoxy(5,10);write('Uhrzeit :',Stunde:2,'
           :',Minute:2,':',sekunde:2);
[CK5]   96:  until keypressed;
[8DK]   97:  Bildschirm1:=Bildschirm2;
              (* Wiederherstellen alten  *)
[CFK]   98:  gotoxy(CursorX,CursorY);
              (* Bildschirm und Cursorpos *)
[TU0]   99:  end;
[761]  100: (*$F- *)
[000]  101:
[000]  102:
[6Y5]  103: (* ---------------------------------  *)
[4TB]  104: (*   Hauptprogramm  bzw Installation  *)
[6Y5]  105: (* ---------------------------------  *)
[000]  106:
[BH1]  107: begin
[LD4]  108:  TurboSS:=Sseg;
[MS4]  109:  TurboSP:=Sptr+4;
[DC7]  110:  IntProg:=@NeuProgramm;
[196]  111:  GetIntVec($16,altint16);
[0EF]  112:  SetIntVec($16,@Int16);              (* Neu
           en Interrupt 16 setzen *)
[NVH]  113:  SetIntVec($1b,SaveInt1b);           (* Bre
           akVektor zurücksetzen  *)
[C2F]  114:  SetIntVec(0,SaveInt00);             (* Int
           0 zurücksetzen         *)
[BS1]  115:  Keep(0);
[5T0]  116: end.
```

**Listing 8. Datum und Uhrzeit auf Tastendruck**

# Ihr Weg zum Erfolg ...

## DMV Zeitschriften

### DOS International
**Die erfolgreiche und allseits beliebte Zeitschrift für alle Anwender von PC's unter MS-DOS**

### PC Amstrad International
**»Die« Nummer Eins, wenn es um CPC's ,PCW Joyce und PC 1512/1640 geht. Im 4. Jahrgang und beliebt wie eh und je.**

### Pascal International
**Keine andere Computerzeitschrift setzt sich so konsequent für moderne Computersprachen ein! Längst mehr als nur ein Blatt für Insider.**

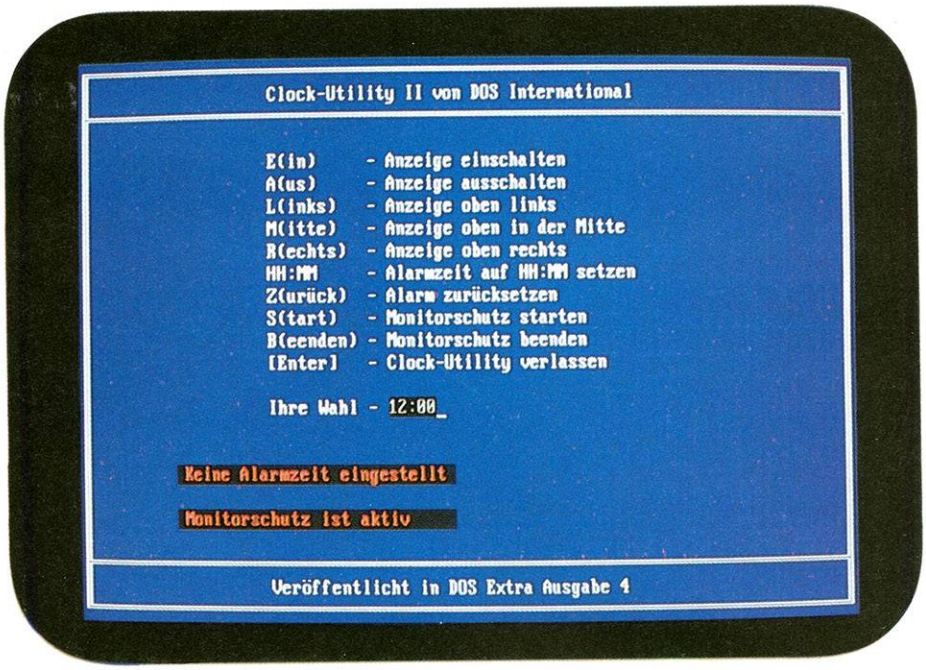
### *Jetzt neu*
### WINDOW
**Das Magazin, das neue Maßstäbe für den Umgang mit anspruchsvoller Software setzt! Von Profis gemacht, nicht nur für Profis gedacht!**

**Ordnung und Übersicht schaffen die beliebten DMV Sammelmappen**

**DMV Verlag · Postfach 250 · 3440 Eschwege**

# Clock-Utility für alle Grafikkarten

**Bereits im ersten DOS Extra war ein Clock-Utility abgedruckt, das permanent eine Uhr auf dem Bildschirm darstellt, das Einstellen einer Alarmzeit erlaubt und die Bildröhre bei längerem Nichtgebrauch zur Schonung abschaltet. Jetzt liegt dieses Programm in einer verbesserten Form vor und ist auch für Hercules-Grafikkarten geeignet.**




**Die Benutzeroberfläche des Clock-Utility listet alle Funktionen des Programms auf**

Schon seit Stunden sitzen Sie vor Ihrem Personalcomputer und haben Raum und Zeit vergessen. Damit letzteres nicht zu häufig passiert, schreibt das Clock-Utility permanent die aktuelle Zeit auf den Bildschirm – wahlweise rechts, links oder in die Mitte der obersten Bildschirmzeile. Damit Sie keine wichtigen Termine versäumen, können Sie auch eine Alarmzeit angeben, zu der fünf aufeinanderfolgende Signaltöne erzeugt werden. Jede volle Stunde und der erste Aufruf des Clock-Utility werden ebenfalls durch einen Signalton angezeigt.

Haben Sie vergessen, Ihren Computer abzuschalten, wenn Sie den Arbeitsplatz verlassen, dann brennt der Elektronenstrahl der Bildröhre das aktuelle Bild in den Schirm ein. Dies wird beim Clock-Utility von einer speziellen Routine verhindert, die automatisch die Bildröhre abschaltet, wenn der Computer drei Minuten lang nicht benutzt wird. Diese Einrichtung funktioniert allerdings nur bei 100 Prozent kompatiblen CGA- und Hercules-Grafikkarten.

Das Clock-Utility arbeitet als residentes Programm vollständig im Hintergrund, so daß die Funktion anderer Programme nicht gestört wird. Die Auswahl der Funktionen des Clock-Utility erfolgt in einem Menü, das nach dem Start des Programms auf dem Bildschirm erscheint (Bild).

Der Quellcode des Clock-Utility besteht aus den zwei Assemblerprogrammen in Listing 1 und 2 und muß von dem Makroassembler MASM in ein ausführbares Programm übersetzt werden.

Nachdem Sie die beiden Listings mit einem Editor, der reinen ASCII-Text erzeugt, eingegeben und ihnen die Dateinamen »clock.asm« und »clkframe.asm« zugewiesen haben, übersetzen Sie Listing 1 mit

```
masm clock ;
```

und Listing 2 mit

```
masm clkframe ;
```

Die beiden erzeugten Objektdateien binden Sie mit dem Linker »link« durch

```
link clock+clkframe,clock ;
```

zu der Datei »clock.exe« zusammen. Abschließend müssen Sie diese Datei nur noch mit

```
exe2bin clock clock.com
```

in die lauffähige COM-Datei umwandeln.

Listing 1 und 2 sind für CGA-, EGA- und VGA-Karten geschrieben. Zur Anpassung an die Hercules-Grafikkarte müssen Sie die Zeilen aus Listing 3 in Listing 1 und die Zeilen aus Listing 4 in Listing 2 ersetzen.

Wenn Sie nicht den Assembler MASM besitzen, können Sie das Clock-Utility auch in der bereits übersetzen Form (Listing 5) mit einem Editor als ASCII-Text eingeben. Dabei ersparen Sie sich sogar viel Tipparbeit. In diesem Fall prüfen Sie mit dem Prüfsummenprogramm »SCheck«, ob Sie alles korrekt eingegeben haben und erzeugen anschließend mit der Option »/b« von SCheck die Binärdatei »clock.bin«, die Sie nur noch mit

```
ren clock.bin clock.com
```

umbenennen müssen, um die lauffähige Version des Clock-Utility zu erhalten.

| Zeile | Spalte | alter Wert (hex) | neuer Wert (hex) |
|---|---|---|---|
| 4 | 16 | 03 | 07 |
| 5 | 1 | 7F | 75 |
| 13 | 12 | 4F | 70 |
| 16 | 2 | CF | F0 |
| 19 | 9 | 03 | 07 |
| 19 | 10 | 7F | 75 |
| 48 | 13 | 0C | 0F |
| 48 | 14 | 01 | 00 |
| 78 | 9 | 17 | 0F |
| 78 | 10 | 01 | 00 |
| 80 | 2 | 1F | 0F |
| 80 | 3 | 01 | 00 |
| 81 | 1 | 0C | 0F |
| 81 | 2 | 01 | 00 |
| 81 | 8 | 1F | 0F |
| 81 | 9 | 01 | 00 |
| 82 | 7 | 0C | 0F |
| 82 | 8 | 01 | 00 |
| 82 | 14 | 0C | 07 |
| 140 | 1 | 07 | 0B |
| 140 | 2 | 06 | 0A |
| 140 | 11 | 1F | 07 |
| 141 | 4 | 1F | 07 |
| 141 | 5 | 01 | 00 |
| 141 | 8 | 00 | 01 |
| 143 | 6 | 95 | 93 |
| 146 | 8 | 00 | 01 |
| 150 | 3 | 01 | 02 |
| 150 | 8 | 1F | 07 |
| 150 | 9 | 01 | 00 |
| 153 | 11 | 13 | 12 |
| 154 | 15 | 1F | 07 |
| 154 | 16 | 01 | 00 |

**Tabelle. Diese Werte müssen Sie in Listing 5 ändern, um das Clock-Utility an die Hercules-Grafikkarte anzupassen**

Wenn Sie auch das Programm »SCheck« nicht besitzen, können Sie Listing 5 mit dem MS-DOS-Debugger »debug« eingeben. Diese Eingabe gestaltet sich etwas mühsam – ist aber immer noch besser, als auf »clock« zu verzichten. Starten Sie also »debug« und legen Sie den Inhalt des Registers CX mit

```
r cx
09C4
```

auf den Wert 09C4hex fest. Dieser Wert gibt die Dateilänge für das Clock-Utility an.
Dann geben Sie durch

```
n clock.com
```

den Namen der Datei an.
Mit dem Befehl »e« geben Sie nun Listing 5 zeilenweise ein, verzichten aber auf die Eingabe der den 16 Hexadezimalwerten vorausgehenden Prüfsumme und Zeilennummer. Die 16 Werte der ersten Zeile geben Sie mit

```
e 0100 e9 45 02 01 00 00 00 00 00 cd ab cd ab 07 00 20
```

ein. Zwischen jedem Wert muß immer genau ein Leerzeichen stehen. Die zweite Zeile geben Sie nach dem gleichen Schema mit

```
e 0110 20 20 20 3a 20 20 3a 20 20 20 20 fa 50 1e 9c 0e
```

ein. Für die dritte Zeile starten Sie mit

```
e 0120 . . . . .
```

für die vierte Zeile mit

```
e 0130 . . . . .
```

und so weiter.

## Selbst mit dem MS-DOS-Debugger »debug« können Sie das Listing des Clock-Utility eingeben

Bei der vierstelligen Zahl direkt hinter dem Befehl »e« handelt es sich immer um die hexadezimale Adresse, ab der die nachfolgenden 16 Werte abgelegt werden. Achten Sie darauf, daß aus diesem Grund auf den Wert 0190hex nicht 0200hex, sondern 01A0hex folgt. Wenn Sie sich einmal verzählt haben, können Sie mit der Formel

```
256 + (16 x zeilennummer)
```

jederzeit aus einer Zeilennummer die zugehörige Adresse für den E-Befehl ermitteln, die Sie allerdings noch in das Hexadezimalsystem umrechnen müssen.
Speichern Sie bei der Eingabe von Listing 5 in den Debugger Ihr Ergebnis regelmäßig mit

```
w
```

auf Diskette oder Festplatte, damit Ihnen Ihre Daten bei einer Fehlbedienung nicht verlorengehen. Haben Sie sämtliche Werte eingegeben, so benutzen Sie wieder

```
w
```

um das komplette Programm zu sichern.
Mit

```
q
```

verlassen Sie den Debugger und können sich davon überzeugen, daß sich die 2500 Byte lange Datei »clock.com« auf Ihrer Diskette beziehungsweise Festplatte befindet. Falls Sie keinen Fehler bei der Eingabe gemacht haben, läuft das Programm korrekt.
Haben Sie sich vertippt, so müssen Sie das Clock-Utility mit

```
debug clock.com
```

wieder in den Debugger laden und sich die Hexadezimalwerte mit

```
d 0100
```

anschauen und mit Listing 5 vergleichen. Allerdings müssen Sie nicht jedesmal für den Befehl »d« eine Adresse angeben, für die nachfolgenden Datenzeilen reicht

```
d
```

aus. Finden Sie einen Fehler, so müssen Sie die Zeile mit

```
e <adresse> <wert1, wert2, ... wert16>
```

korrigieren. Wer sich bereits besser mit dem Debugger auskennt, kann die Daten auch eleganter korrigieren, ohne die ganze Zeile von neuem eingeben zu müssen. Schauen Sie dazu in Ihre MS-DOS-Handbuch.
Bei Listing 5 handelt es sich übrigens wie bei Listing 1 und 2 ebenfalls um die Farbversion des Clock-Utility. Doch für die Anpassung an eine Hercules-Grafikkarte müssen Sie nur die 33 hexadezimalen Werte gemäß den Angaben in der Tabelle ändern.
Zu guter Letzt noch ein wichtiger Hinweis zur Arbeit mit dem Clock-Utility. Wenn Sie dieses Programm in Ihre Autoexec-Datei aufnehmen und Ihnen das Einstellen der Parameter zu umständlich ist, können Sie sich eine kleine Setup-Datei schreiben, die über eine Eingabeumleitung das Clock-Utility beim Aufruf automatisch so einrichtet, wie Sie es sich wünschen. Definieren Sie beispielsweise eine Setup-Datei »clock.set« mit

```
copy con clock.set
```

und geben Sie folgendes ein:

```
e
r
s
```

(Vergessen Sie nicht die Leerzeile am Schluß der Datei, damit das Clock-Utility korrekt beendet wird.)
Wenn Sie nun in Ihre Autoexec-Datei den Aufruf des Clock-Utility mit

```
clock < clock.set
```

eintragen, so wird beim Booten des PC »clock« gestartet, durch das »e« in der Setp-Datei wie von Geisterhand die Uhr eingeschaltet, die Uhr durch das »r« in die rechte obere Ecke gesetzt, mit »s« der Monitorschutz aktiviert und durch die Leerzeile das Programm anschließend wieder verlassen.

(J. Bauer/hg/ma)

```
[045]    0: title clock.asm
[462]    1: page  66,132
[V10]    2: ;
[DV4]    3: ;*********************************
[DS3]    4: ;*                                 *
[4C9]    5: ;*  »clock.com« wird mit           *
[XV7]    6: ;*     masm clkframe.asm           *
[9S6]    7: ;*     masm clock.asm              *
[EV9]    8: ;*     link clock+clkframe,clock   *
[1C8]    9: ;*     exe2bin clock clock.com     *
[7N5]   10: ;*  erzeugt                         *
[DS3]   11: ;*                                 *
[DV4]   12: ;*********************************
[V10]   13: ;
[S18]   14: cseg            segment public 'code'
[DL1]   15:                 org 100h
[V10]   16: ;
[9F9]   17: public          module2,get,cl_com,as1
[ERH]   18: extrn           pos_cur:near,dloop:near,exit:
            near,module1:near
```

```
[V10]   19: ;
[7LF]   20: vector          equ 1ch*4       ;Vector table
            offset of int 1ch
[FP9]   21: asi_segm        equ 40h         ;ASI segment
[44D]   22: vector_segm     equ 00h         ;Vector table
            segment
[V10]   23: ;
[7L4]   24: addr_6845       equ 63h
[C95]   25: crt_mode        equ 65h
[R15]   26: tick_30         equ 1c8h
[V10]   27: ;
[73E]   28: assume          cs:cseg,ds:cseg,es:cseg,ss:no
            thing
[V10]   29: ;
[HW5]   30: module2         proc far
[VE4]   31:                 jmp transient
[V10]   32: ;
[753]   33: busy            db 1
```

```
[PE3]   34: ticks          db 0
[5T2]   35: scp            dw 0
[KC4]   36: d_c_pos        dw 0
[0X6]   37: alrm_time      dw 0abcdh
[0E2]   38:                dw 0abcdh
[BR4]   39: alrm_flag      db 7,0
[XH5]   40: alrm_disp      dw 2020h
[5M5]   41: tubehour       dw 2020h
[G11]   42:                db ':'
[9K5]   43: tubemin        dw 2020h
[G11]   44:                db ':'
[2G5]   45: tubesec        dw 2020h
[6F1]   46:                dw 2020h
[V10]   47: ;
[BA4]   48: ;*****************************
[V10]   49: ;
[A3H]   50: resident:                      ;Start of inte
          rrupt service routine
[6C8]   51:                cli             ;Disable inter
          rupts
[1M2]   52:                push ax
[GL2]   53:                push ds
[HK1]   54:                pushf
[AL2]   55:                push cs
[R3B]   56:                pop ds          ;Get correct d
          ata segment
[7M2]   57:                push bx
[DM2]   58:                push cx
[LM2]   59:                push dx
[4M2]   60:                push si
[AH2]   61:                push di
[FK2]   62:                push bp
[PL2]   63:                push es
[255]   64:                mov ax,asi_segm
[4C2]   65:                mov es,ax
[CM7]   66:                cmp byte ptr[ticks],12h
[VD9]   67:                jz activate     ;Jump every se
          cond
[VGF]   68:                inc byte ptr[ticks] ;Incremen
          t 1/18 sec counter
[LHG]   69:                jmp recover_regs    ;Jump to
          end of interrupt service
[V10]   70: ;
[113]   71: activate:
[BUE]   72:                mov byte ptr[ticks],00   ;Res
          et 1/18 sec counter
[T1C]   73:                cmp byte ptr[busy],0     ;Che
          ck busy flag
[4K4]   74:                                         ;(e.
          g. still alarm)
[2R3]   75:                jnz get_time
[F38]   76: chk_al:        jmp check_alarm
[V10]   77: ;
[SZ2]   78: get_time:
[CF2]   79:                mov ah,0fh
[E02]   80:                int 10h
[RMD]   81:                cmp al,3        ;Check video m
          ode is not graphics
[LN2]   82:                jg chk_al
[E3F]   83:                mov ax,word ptr es:[006eh]
          ;Address of 'DOS' seconds
[S62]   84:                mov cl,4
[DM2]   85:                push cx
[WA2]   86:                shl ax,cl
[30K]   87:                mov bx,word ptr es:[006ch]
          ;Address of 'DOS' minutes and hours
[0BE]   88:                mov dx,bx       ;Convert time
          in 'DOS' format to ASCII
[LME]   89:                mov cl,0ch      ;formatted tim
          e, for being displayed
[XB2]   90:                shr dx,cl
[5A2]   91:                add dx,ax
[EC2]   92:                mov ax,bx
[X52]   93:                pop cx
[WA2]   94:                shl ax,cl
[TZ2]   95:                mov bx,4446h
[KY1]   96:                div bx
[CT7]   97:                mov word ptr [tubesec],dx
[EK2]   98:                mov bx,3ch
[WC2]   99:                xor dx,dx
[KY1]  100:                div bx
[1V7]  101:                mov word ptr [tubemin],dx
[UL3]  102:                call convert
[368]  103:                cmp ax,word ptr [tubehour]
[RN5]  104:                jnz signal_hour
[2PA]  105: cont:          mov word ptr [tubehour],ax
[808]  106:                mov ax,word ptr [tubemin]
[UL3]  107:                call convert
[1V7]  108:                mov word ptr [tubemin],ax
[G08]  109:                mov ax,word ptr [tubesec]
[NY1]  110:                mul bx
[TZ2]  111:                mov bx,4446h
[KY1]  112:                div bx
[UL3]  113:                call convert
[CT7]  114:                mov word ptr [tubesec],ax
[LRA]  115:                jmp display_on ;Jump to displ
          ay
[Y56]  116: signal_hour:   push ax
[0H5]  117:                call speaker_on
```

```
[LY1]  118:                pop ax
[ZM2]  119:                jmp cont
[V10]  120: ;
[E47]  121: convert        proc near
[AL0]  122:                aam
[G6C]  123:                add ax,3030h    ;Convert decim
          al to ASCII and
[Y3C]  124:                xchg ah,al      ;exchange high
          and low nibble
[SM0]  125:                ret
[VP4]  126: convert        endp
[V10]  127: ;
[TF3]  128: display_on:
[P52]  129:                mov ah,3
[292]  130:                xor bh,bh
[7H9]  131:                int 10h         ;Fetch cursor
          position
[7NG]  132:                mov word ptr[scp],dx     ;Sav
          e current cursor position
[ZE5]  133:                mov dx,cs:[d_c_pos]
[P52]  134:                mov ah,2
[0W8]  135:                int 10h         ;Position curs
          or
[XTG]  136:                mov si,offset alrm_disp+1
           ;Address start of display
[3P8]  137:                mov cx,0ah      ;Length of dis
          play
[MHB]  138: diloop:        mov al,[si]     ;Load characte
          r
[4M2]  139:                push si
[DM2]  140:                push cx
[P52]  141:                mov ah,9
[092]  142:                mov cx,1
[9G6]  143:                mov bl,04fh     ;Attribute
[XFC]  144:                int 10h         ;Write charact
          er and attribute
[VRB]  145:                inc dl          ;Increment cur
          sor position
[P52]  146:                mov ah,2
[292]  147:                xor bh,bh
[0W8]  148:                int 10h         ;Position curs
          or
[XY1]  149:                pop cx
[LY1]  150:                pop si
[WPA]  151:                inc si          ;Address next
          character
[B63]  152:                loop diloop
[PSG]  153:                cmp byte ptr [alrm_disp],0eh
           ;Check for alarm display
[GR3]  154:                jnz rec_curs
[ZE5]  155:                mov dx,cs:[d_c_pos]
[2B2]  156:                xor bh,bh
[P52]  157:                mov ah,2
[E02]  158:                int 10h
[AS6]  159:                mov ax,90eh     ;Load symbol
[0K9]  160:                mov bl,0cfh     ;Blinking attr
          ibute
[092]  161:                mov cx,1
[E02]  162:                int 10h
[V3N]  163: rec_curs:      mov dx,word ptr[scp]     ;Rec
          over former cursor position
[P52]  164:                mov ah,2
[292]  165:                xor bh,bh
[E02]  166:                int 10h
[UK3]  167: check_alarm:
[B1H]  168:                cmp byte ptr[alrm_flag],5
           ;Check for alarm finished
[RCC]  169:                jb give_alarm  ;Jump if not f
          inished yet
[F78]  170:                mov ax,word ptr[alrm_time]
[M4H]  171:                cmp ax,word ptr[tubehour]
           ;Check for alarm hour reached
[KS9]  172:                jnz recover_regs    ;Jump if
          not
[NM8]  173:                mov ax,word ptr[alrm_time+2]
[KMK]  174:                cmp ax,word ptr[tubemin] ;Che
          ck for alarm minute reached
[KS9]  175:                jnz recover_regs    ;Jump if
          not
[0VF]  176:                mov word ptr [alrm_flag],0
           ;Reset alarm count flag
[V10]  177: ;
[Y93]  178: give_alarm:
[K63]  179:                call buzzer
[V10]  180: ;
[2U7]  181: recover_regs   proc near
[CF2]  182:                mov ah,0fh
[E02]  183:                int 10h
[E52]  184:                cmp al,3
[TW2]  185:                jg end_int
[4W3]  186:                cmp cs:delay,0
[ZX2]  187:                jz end_int
[836]  188:                call check_keypress
[2F5]  189:                cmp cs:video_flag,0
[ZX2]  190:                jz end_int
[1C3]  191:                dec cs:count
[Y83]  192:                jnz end_int
[CP5]  193:                mov cs:count,tick_30
[XH4]  194:                dec cs:timeout
```

```
[Y83]  195:                jnz end_int
[615]  196:                mov al,cs:delay
[L15]  197:                mov cs:timeout,al
[UP4]  198:                call video_off
[V10]  199: ;
[H85]  200: end_int:       pop es
[8X1]  201:                pop bp
[8W1]  202:                pop di
[LY1]  203:                pop si
[2Z1]  204:                pop dx
[XY1]  205:                pop cx
[SY1]  206:                pop bx
[P11]  207:                popf
[4Y1]  208:                pop ds
[LY1]  209:                pop ax
[PM0]  210:                sti
[C21]  211:                iret
[FC5]  212: recover_regs   endp
[V10]  213: ;
[3DH]  214: check_keypress proc near      ;Check for key
       was pressed
[XCF]  215:                mov ax,word ptr cs:[scp] ;Res
       et counters if so
[289]  216:                cmp ax,word ptr cs:[last_cur]
[L66]  217:                mov cs:[last_cur],ax
[1U2]  218:                jz no_key
[SF3]  219: keypressed:
[Y64]  220:                mov si,addr_6845
[FP7]  221:                mov dx,word ptr es:[si]
[D62]  222:                add dx,4
[745]  223:                mov si,crt_mode
[9M7]  224:                mov al,byte ptr es:[si]
[TC2]  225:                out dx,al
[5G5]  226:                mov cs:video_flag,1
[CP5]  227:                mov cs:count,tick_30
[615]  228:                mov al,cs:delay
[L15]  229:                mov cs:timeout,al
[5E5]  230: no_key_pressed:
[SM0]  231:                ret
[UL2]  232: no_key:
[0F4]  233:                mov ax,es:[17h]
[ML5]  234:                cmp ax,last_kb_flag
[3K7]  235:                mov cs:[last_kb_flag],ax
[AX4]  236:                jnz keypressed
[WN4]  237:                mov ax,es:[1ch]
[V85]  238:                cmp ax,last_ptr
[U86]  239:                mov cs:[last_ptr],ax
[AX4]  240:                jnz keypressed
[BW5]  241:                jmp no_key_pressed
[6Y5]  242: check_keypress endp
[V10]  243: ;
[6RE]  244: video_off      proc  near     ;Switch beam c
       urrent off
[Y64]  245:                mov si,addr_6845
[FP7]  246:                mov dx,word ptr es:[si]
[D62]  247:                add dx,4
[745]  248:                mov si,crt_mode
[9M7]  249:                mov al,byte ptr es:[si]
[PM2]  250:                and al,0f7h
[TC2]  251:                out dx,al
[2G5]  252:                mov cs:video_flag,0
[SM0]  253:                ret
[3L4]  254: video_off      endp
[V10]  255: ;
[P7D]  256: delay          db 6           ;Data needed f
       or tube saver
[PG4]  257: timeout        db 2
[RW3]  258: count          dw 456
[LC4]  259: last_cur       dw 0
[VZ4]  260: last_kb_flag   dw 0
[UE4]  261: last_ptr       dw 0
[4F4]  262: video_flag     db 1
[V10]  263: ;
[V10]  264: ;
[1M6]  265: buzzer         proc near
[ZFK]  266:                mov word ptr[alrm_time],0abcd
       h ;Clear alarm time buffer
[EE2]  267:                mov al,20h
[P3F]  268:                mov byte ptr [alrm_disp],al
        ;Clear alarm display
[V10]  269: ;
[KC3]  270: speaker_on:
[R7A]  271:                mov al,3h      ;Set up first
       frequency
[AX2]  272:                call sound
[X7H]  273:                inc byte ptr[alrm_flag+1]
        ;Set first frequency is done
[7SA]  274:                mov al,4h      ;Set up second
       frequency
[AX2]  275:                call sound
[RBH]  276:                inc byte ptr[alrm_flag+1]
        ;Set second frequency is done
[15A]  277:                mov al,5h      ;Set up third
       frequency
[AX2]  278:                call sound
[P9H]  279:                inc byte ptr[alrm_flag]  ;Inc
       rement alarm count flag
[UBH]  280:                mov byte ptr[alrm_flag+1],0
        ;Set third frequency is done
```

```
[GH4]  281: sp_exit:       ret
[000]  282:
[FY4]  283: sound:         push cx
[1M2]  284:                push ax
[MP2]  285:                mov al,0b6h
[WB2]  286:                out 43h,al
[T92]  287:                xor al,al
[PB2]  288:                out 42h,al
[LY1]  289:                pop ax
[PB2]  290:                out 42h,al
[F12]  291:                in al,61h
[1M2]  292:                push ax
[0Y1]  293:                or al,3
[UB2]  294:                out 61h,al
[4T3]  295:                mov cx,0ffffh
[E62]  296:                mov al,2
[2E8]  297: soundloop:     loop soundloop
[493]  298:                mov cx,0fffh
[4U1]  299:                dec al
[E52]  300:                cmp al,1
[CY3]  301:                jz soundloop
[LY1]  302:                pop ax
[UB2]  303:                out 61h,al
[XY1]  304:                pop cx
[SM0]  305:                ret
[E64]  306: buzzer         endp
[V10]  307: ;
[BA4]  308: ;*****************************
[V10]  309: ;
[L43]  310: transient:
[AL2]  311:                push cs
[R3B]  312:                pop ds         ;Get correct d
       ata segment
[AY8]  313:                call module1   ;Call the AS f
       rame
[XZ4]  314:                mov si,offset msg1
[53A]  315:                call dloop     ;Display main
       message
[913]  316:                mov dx,0f14h
[MU2]  317:                call dloop
[RZ2]  318:                mov dx,0414h
[MU2]  319:                call dloop
[SZ2]  320:                mov dx,0514h
[MU2]  321:                call dloop
[TZ2]  322:                mov dx,0614h
[MU2]  323:                call dloop
[UZ2]  324:                mov dx,0714h
[MU2]  325:                call dloop
[VZ2]  326:                mov dx,0814h
[MU2]  327:                call dloop
[WZ2]  328:                mov dx,0914h
[MU2]  329:                call dloop
[413]  330:                mov dx,0a14h
[MU2]  331:                call dloop
[513]  332:                mov dx,0b14h
[MU2]  333:                call dloop
[613]  334:                mov dx,0c14h
[MU2]  335:                call dloop
[713]  336:                mov dx,0d14h
[MU2]  337:                call dloop
[P33]  338:                mov ax,351ch
[V0B]  339:                int 21h        ;Get vector of
        interrupt 1C
[TWL]  340:                cmp byte ptr es:[bx],0cfh
        ;Check for CLOCK already being installed
[D47]  341:                jz install     ;Jump if not
[E24]  342:                call parm_inp
[LP2]  343:                call exit
[72B]  344:                int 20h        ;Finish transi
       ent program
[V10]  345: ;
[2P0]  346: as1:
[BCE]  347:                db '                    Veröff
       entlicht in DOS Extra Ausgabe 4',0
[V10]  348: ;
[FY2]  349: install:
[SH5]  350:                mov bx,vector_segm
[XVC]  351:                mov es,bx      ;Set ES to seg
       ment of vector table
[EC2]  352:                mov bx,ds
[6C8]  353:                cli            ;Disable inter
       rupts
[LF4]  354:                mov si,vector+2
[2CN]  355:                mov word ptr es:[si],bx  ;Ins
       tall new interrupt segment vector
[CW1]  356:                dec si
[CW1]  357:                dec si
[Z17]  358:                mov bx,offset resident
[U6M]  359:                mov word ptr es:[si],bx  ;Ins
       tall new interrupt offset vector
[4P7]  360:                sti            ;Enable interr
       upts
[9C2]  361:                mov ax,ds
[R8C]  362:                mov es,ax      ;Set ES to cur
       rent data segment
[2C9]  363:                call parm_inp  ;Get parameter
       s
[LP2]  364:                call exit
[7V7]  365:                mov dx,offset transient
```

```
[9BG]  366:                inc dx          ;Terminate pro
       gram and let stay resident
[PPH]  367:                int 27h         ;all coding, t
       hat is before the label 'transient'
[9Z6]  368: parm_inp       proc near
[2TD]  369:                mov word ptr [dloop+1],010ch
        ;Set up red on black
[SF6]  370:                call show_alarm_time
[743]  371:                call cl_com
[EK3]  372:                call cl_disp
[6F2]  373:                call get
[DC2]  374:                xor ax,ax
[4S8]  375:                xor bx,bx       ;Clear registe
       rs
[YB2]  376:                xor si,si
[792]  377:                xor ch,ch
[0RB]  378:                mov di,80h      ;Address of pa
       rameter count
[REC]  379:                mov cl,byte ptr[di] ;Paramete
       r count to CL
[0FM]  380:                mov bp,offset alrm_hour-1
        ;Address of transient alarm time buffer
[S52]  381:                cmp cl,0
[BA4]  382:                jnz next_parm
[5YF]  383:                jmp go_exit     ;Jump if no pa
       rameters in command line
[L33]  384: next_parm:
[VNA]  385:                inc di          ;Address next
       parameter
[86A]  386:                mov bl,byte ptr[di] ;Paramete
       r to BL
[LD2]  387:                cmp bl,40h
[9B1]  388:                jl go
[LD2]  389:                cmp bl,50h
[BM2]  390:                jge lu_c1
[122]  391:                or bl,60h
[HX1]  392:                jmp go
[VV4]  393: lu_c1:         cmp bl,60h
[4W1]  394:                jge go
[822]  395:                or bl,70h
[RG4]  396: go:            cmp bl,'e'
[6T2]  397:                jnz l_top
[9V7]  398:                mov byte ptr es:[busy],1
[W33]  399:                mov dx,140ah
[705]  400:                mov si,offset msg6
[MU2]  401:                call dloop
[3R3]  402:                jmp parm_inp
[EY1]  403: l_top:
[DB2]  404:                cmp bl,'l'
[BT2]  405:                jnz m_top
[UPB]  406:                mov es:[d_c_pos],0  ;left han
       d display
[W33]  407:                mov dx,140ah
[905]  408:                mov si,offset msg7
[MU2]  409:                call dloop
[3R3]  410:                jmp parm_inp
[FY1]  411: m_top:
[EB2]  412:                cmp bl,'m'
[4U2]  413:                jnz r_top
[RL6]  414:                mov es:[d_c_pos],0023h
[W33]  415:                mov dx,140ah
[B05]  416:                mov si,offset msg8
[MU2]  417:                call dloop
[3R3]  418:                jmp parm_inp
[MY1]  419: r_top:
[LB2]  420:                cmp bl,'r'
[FS3]  421:                jnz tubesave
[76D]  422:                mov es:[d_c_pos],0046h   ;rig
       ht hand display
[W33]  423:                mov dx,140ah
[D05]  424:                mov si,offset msg9
[MU2]  425:                call dloop
[3R3]  426:                jmp parm_inp
[H03]  427: tubesave:
[MB2]  428:                cmp bl,'s'
[S74]  429:                jnz tubes_off
[3T4]  430:                mov es:[delay],6
[W33]  431:                mov dx,140ah
[D45]  432:                mov si,offset msg10
[MU2]  433:                call dloop
[3R3]  434:                jmp parm_inp
[M33]  435: tubes_off:
[3B2]  436:                cmp bl,'b'
[8X2]  437:                jnz cancel
[3T4]  438:                mov es:[delay],0
[W33]  439:                mov dx,140ah
[G45]  440:                mov si,offset msg11
[MU2]  441:                call dloop
[3R3]  442:                jmp parm_inp
[KG2]  443: cancel:
[2B2]  444:                cmp bl,'a'
[BHC]  445:                jnz is_it_reset     ;Jump if
       not 'clear'
[2GK]  446:                mov byte ptr es:[busy],0 ;Res
       et busy flag in resident part
[W33]  447:                mov dx,140ah
[105]  448:                mov si,offset msg3
[KDD]  449:                call dloop      ;Display 'swit
       ched off' message
[3R3]  450:                jmp parm_inp
[DT3]  451: is_it_reset:
[VB2]  452:                cmp bl,'z'
[SN2]  453:                jnz blank
[72N]  454:                mov word ptr es:[alrm_time],0
       abcdh ;Clear resident alarm buffer
[5DH]  455:                mov byte ptr es:[alrm_disp],'
       '     ;Clear alarm display
[3R3]  456:                jmp parm_inp
[GZ2]  457: go_exit:
[TM3]  458:                jmp end_parm
[8W1]  459: blank:
[192]  460:                cmp bl,' '
[2CC]  461:                jnz dpoint_check    ;Jump if
       not 'space'
[W25]  462: get_next_parm:
[RX2]  463:                loop l_n_p
[9ZK]  464:                cmp bp,offset alrm_hour-1
        ;Check if alarm time was requested
[PU2]  465:                jnz nxt
[6F7]  466:                jmp parm_inp    ;Jump if not
[5X6]  467: l_n_p:         jmp next_parm
[2DN]  468: nxt:           cmp bp,offset alrm_min+1 ;Che
       ck for alarm time request was HH:MM
[1ME]  469:                jnz alrm_err    ;Jump if alarm
        time is incorrect
[7F8]  470:                mov ax,word ptr [alrm_hour]
[9S2]  471:                xchg ah,al
[3G2]  472:                cmp ax,'23'
[LNG]  473:                jg alrm_err     ;Jump if reque
       sted alarm hour is not OK
[268]  474:                mov ax,word ptr [alrm_min]
[9S2]  475:                xchg ah,al
[KG2]  476:                cmp ax,'59'
[T1H]  477:                jng store_alarm      ;Jump if
       requested alarm minute is OK
[MY2]  478: alrm_err:
[L67]  479:                mov si,offset err_msg_1
[W33]  480:                mov dx,140ah
[87C]  481:                call dloop      ;Display 'wron
       g time' message
[3R3]  482:                jmp parm_inp
[YN3]  483: store_alarm:
[GXD]  484:                mov bp,word ptr [alrm_hour]
        ;Alarm hour to BP
[YAM]  485:                mov word ptr es:[alrm_time],b
       p     ;Alarm hour to resident buffer
[LYD]  486:                mov bp,word ptr [alrm_min]
        ;Alarm minute to BP
[X3N]  487:                mov word ptr es:[alrm_time+2]
       ,bp   ;Alarm minute to resident buffer
[ERG]  488:                mov byte ptr es:[alrm_disp],0
       eh    ;Set alarm display
[305]  489:                mov si,offset msg4
[W33]  490:                mov dx,140ah
[G1D]  491:                call dloop      ;Display 'alar
       m stored' message
[3R3]  492:                jmp parm_inp
[V10]  493: ;
[N64]  494: dpoint_check:
[V92]  495:                cmp bl,':'
[0V7]  496:                jz dpoint1      ;Jump if not '
       :'
[F92]  497:                cmp bl,'.'
[XT7]  498:                jnz cipher      ;Jump if not '
       .'
[B6P]  499: dpoint1:       cmp bp,offset alrm_hour+1
        ;Check for two digits were before ':'
[34G]  500:                jnz alrm_er1    ;Jump if not t
       wo digits for alarm hour
[CKC]  501:                cmp cx,3        ;Check for 3 p
       arameters minimum
[N4C]  502:                jb alrm_err     ;Jump if less
       parameters
[MLD]  503:                inc bp          ;Address next
       field of alarm buffer
[0N5]  504:                jmp get_next_parm
[MY2]  505: alrm_er1:
[GFH]  506:                cmp bp,offset alrm_hour  ;Che
       ck for one digit for hour
[9G7]  507:                jnz alrm_err    ;Jump if not
[7M2]  508:                push bx
[C58]  509:                mov bl,byte ptr[alrm_hour]
[A09]  510:                mov byte ptr[alrm_hour+1],bl
[8X8]  511:                mov byte ptr[alrm_hour],30h
[SY1]  512:                pop bx
[082]  513:                cmp cx,2
[MB3]  514:                jb alrm_err
[L42]  515:                add bp,2
[0N5]  516:                jmp get_next_parm
[0L2]  517: cipher:
[H92]  518:                cmp bl,'0'
[UR9]  519:                jb err_msg      ;Jump if param
       eter <0
[U92]  520:                cmp bl,'9'
[2T9]  521:                ja err_msg      ;Jump if param
       eter >9
[MLD]  522:                inc bp          ;Address next
       field of alarm buffer
```

```
[7GF]  523:                mov byte ptr [bp],bl     ;Put
         digit into alarm buffer
[0N5]  524:                jmp get_next_parm
[ZX2]  525: err_msg:
[U67]  526:                mov si,offset err_msg_2
[W33]  527:                mov dx,140ah
[D6F]  528:                call dloop     ;Display 'inva
         lid option' message
[3R3]  529:                jmp parm_inp
[9E4]  530: end_parm:      ret
[PH4]  531: parm_inp       endp
[V10]  532: ;
[ED8]  533: show_alarm_time proc near
[8V9]  534:                mov ax,word ptr es:[alrm_time
         ]
[WP3]  535:                cmp ax,0abcdh
[YDC]  536:                jz not_set     ;Jump if no al
         arm time is set
[D79]  537:                mov word ptr [alrm_hour+2],':
         '
[UCE]  538:                mov word ptr [alrm_hour],ax
         ;Put contents of
[GFF]  539:                mov ax,word ptr es:[alrm_time
         +2]   ;resident buffer
[ZHE]  540:                mov word ptr [alrm_min],ax
         ;into alarm buffer
[U33]  541:                mov dx,120ah
[XM7]  542:                mov si,offset alarm_time
[1XG]  543:                call dloop     ;Display 'alar
         m time' of resident buffer
[SM0]  544:                ret
[DZ2]  545: not_set:
[UP3]  546:                mov dx,120ah
[505]  547:                mov si,offset msg5
[93D]  548:                call dloop     ;Display 'no a
         larm set' message
[SM0]  549:                ret
[5U6]  550: show_alarm_time endp
[V10]  551: ;
[7D5]  552: get            proc near
[LZ2]  553:                mov bx,0117h
[Z03]  554:                mov dx,0f20h
[EN3]  555:                call pos_cur
[1H2]  556:                mov bp,7fh
[UR5]  557:                mov byte ptr[bp],7
[NC2]  558:                mov dx,bp
[7F2]  559:                mov ah,0ah
[T02]  560:                int 21h
[1V7]  561:                mov word ptr[dloop+1],011fh
[W33]  562:                mov dx,140ah
[XU6]  563:                mov si,offset cl_c_dat
[MU2]  564:                call dloop
[UU7]  565:                mov word ptr[dloop+1],010ch
[SM0]  566:                ret
[MZ2]  567: get            endp
[V10]  568: ;
[CZC]  569: cl_com         proc near      ;Clear command
         line
[HBC]  570:                mov word ptr[dloop+1],011fh
         ;white on blue
[Z03]  571:                mov dx,0f20h
[U57]  572:                mov si,offset cl_c_dat+10
[MU2]  573:                call dloop
[WGB]  574:                mov word ptr[dloop+1],010ch
         ;red on black
[SM0]  575:                ret
[414]  576: cl_com         endp
[V10]  577: ;
[F6E]  578: cl_disp        proc near      ;Clear message
         display
[592]  579:                mov dx,0
[M33]  580:                mov bx,000ch
[EN3]  581:                call pos_cur
[GA2]  582:                mov cx,80
[3Z2]  583:                mov ax,0920h
[E02]  584:                int 10h
[SM0]  585:                ret
[YK4]  586: cl_disp        endp
[884]  587: dummy          db ' ',0
[MN7]  588: cl_c_dat       db '
             ',0
[91K]  589: msg1           db '                    Clock-Ut
         ility II von DOS International',0
[496]  590: msg2           db 'Ihre Wahl - ',0
[TCA]  591:                db 'E(in)      - Anzeige einsc
         halten        ',0
[WFA]  592:                db 'A(us)      - Anzeige aussc
         halten        ',0
[P5A]  593:                db 'L(inks)    - Anzeige oben
         links         ',0
[MNB]  594:                db 'M(itte)    - Anzeige oben
         in der Mitte ',0
[TNA]  595:                db 'R(echts)   - Anzeige oben
         rechts        ',0
[T2B]  596:                db 'HH:MM      - Alarmzeit auf
          HH:MM setzen',0
```

```
[LRA]  597:                db 'Z(urück)   - Alarm zurücks
         etzen         ',0
[XFB]  598:                db 'S(tart)    - Monitorschutz
          starten      ',0
[9MB]  599:                db 'B(eenden) - Monitorschutz
          beenden      ',0
[23C]  600:                db '[Enter]    - Clock-Utility
          verlassen    ',0
[4AD]  601: msg3           db ' Clock-Anzeige ausgeschal
         tet ',0
[MXC]  602: msg4           db ' Neue Alarmzeit eingestel
         lt  ',0
[W8D]  603: msg5           db ' Keine Alarmzeit eingeste
         llt ',0
[09D]  604: msg6           db ' Clock-Anzeige eingeschal
         tet ',0
[DUA]  605: msg7           db ' Anzeige oben links
                ',0
[TRB]  606: msg8           db ' Anzeige oben in der Mitt
         e    ',0
[X8B]  607: msg9           db ' Anzeige oben rechts
                ',0
[HZB]  608: msg10          db ' Monitorschutz ist aktiv
                ',0
[CLC]  609: msg11          db ' Monitorschutz ist inakti
         v    ',0
[7SD]  610: err_msg_1      db ' Uhrzeit falsch definiert
         !    ',0
[P7D]  611: err_msg_2      db ' Parameter ist unbekannt
         !    ',0
[7A6]  612: clr            db '
                ',0
[8H9]  613: alarm_time     db ' Alarmzeit ist '
[1M5]  614: alrm_hour      dw 2020h
[G11]  615:                db ':'
[3H5]  616: alrm_min       dw 2020h
[232]  617:                db '           ',0
[V10]  618: ;
[A64]  619: module2        endp
[7M2]  620:                cseg ends
[G13]  621:                end module2
```

**Listing 1. Das Clock-Utility besteht aus dem Assemblerprogramm »clock.asm« ...**

```
[Z65]    0: title clkframe.asm
[642]    1: page 66,132
[V10]    2: ;
[ULF]    3: ;This module is got to become linked with c
           lock.obj
[V10]    4: ;
[5N7]    5:                 cseg segment public 'code'
[4NC]    6: assume          cs:cseg,ds:cseg,es:cseg,ss:cs
           eg
[V10]    7: ;
[RXB]    8: public          pos_cur,dloop,exit,module1
[U1G]    9: extrn           module2:near,get:near,cl_com:
           near,as1:near
[000]   10:
[R5P]   11: attr            equ 011fh      ;White on blue
            increased intensity on display page 2
[21F]   12: block           equ 0607h      ;Block cursor
           top line 7, bottom 8
[45F]   13: hori            equ 0cdh       ;Horizontal fr
           aming character
[K0E]   14: vert            equ 0bah       ;Vertical fram
           ing character
[2TD]   15: topleft         equ 0c9h       ;Top left fram
           ing corner
[50G]   16: botleft         equ 0c8h       ;Bottom left f
           raming character
[0CG]   17: topright        equ 0bbh       ;Top right fra
           ming character
[LKG]   18: botright        equ 0bch       ;Bottom right
           framing character
[P7C]   19: llink           equ 0cch       ;Left frame li
           nker
[DVB]   20: rlink           equ 0b9h       ;Right frame l
           inker
[VYG]   21: line            equ 4eh        ;Length of hor
           izontal display line
[VVG]   22: lind            equ 80h        ;Address of le
           ngth indicator input
[V10]   23: ;
[V10]   24: ;
[WG2]   25: module1:
[WE4]   26:                 mov bh,0       ;Page 1
```

```
[E59]   27:                mov ah,3        ;Get cursor po
          sition
[E02]   28:                int 10h
[44H]   29:                mov word ptr[cur],dx     ;Sav
          e cursor position from page 1
[KV6]   30:                mov ax,0501h    ;Select page 2
[E02]   31:                int 10h
[NNB]   32:                mov cx,block    ;Set block sty
          le cursor
[P52]   33:                mov ah,1
[E02]   34:                int 10h
[847]   35:                mov ax,0600h    ;Blank window
[B4C]   36:                mov bh,1fh      ;Attribute for
           blanked window
[3AD]   37:                xor cx,cx       ;Upper left co
          rner of window is 0,0
[SGE]   38:                mov dx,1850h    ;Lower right c
          orner of window is 24,80
[E02]   39:                int 10h
[0EB]   40:                mov bx,attr     ;Set up page a
          nd attribute
[C57]   41:                mov dx,0001h    ;Row 0, column
           1
[EN3]   42:                call pos_cur
[1TA]   43:                mov cl,line     ;Set up charac
          ter count
[2H9]   44:                mov al,hori     ;Horizontal fr
          aming
[MS6]   45:                int 10h         ;Frame in row
          0
[CT2]   46:                call skip2
[CT6]   47:                int 10h         ;Frame in row
          4
[BE2]   48:                mov dh,16h
[EN3]   49:                call pos_cur
[Y37]   50:                int 10h         ;Frame in row
          22
[CT2]   51:                call skip2
[C47]   52:                int 10h         ;Frame in row
          24
[TV6]   53:                xor dx,dx       ;Row 0, column
          0
[EN3]   54:                call pos_cur
[EP8]   55:                mov cl,1        ;New character
           count
[H7A]   56:                mov al,topleft ;Top left char
          acter
[E02]   57:                int 10h
[A38]   58:                call nextline   ;Row 1, column
          0
[ZC9]   59:                mov al,vert     ;Vertical char
          acter
[1M2]   60:                push ax
[E02]   61:                int 10h
[F38]   62:                call nextline   ;Row 2, column
          0
[01B]   63:                mov al,llink    ;Left linking
          character
[E02]   64:                int 10h
[YWB]   65:                pop ax          ;Restore verti
          cal character
[1M2]   66:                push ax
[UN2]   67:                call loop
[5K8]   68:                call nextline   ;Row 22, colum
          n 0
[283]   69:                mov al,llink
[E02]   70:                int 10h
[BK8]   71:                call nextline   ;Row 23, colum
          n 0
[YWB]   72:                pop ax          ;Restore verti
          cal character
[E02]   73:                int 10h
[HK8]   74:                call nextline   ;Row 24, colum
          n 0
[W6B]   75:                mov al,botleft ;Bottom left c
          haracter
[E02]   76:                int 10h
[W39]   77:                mov dx,004fh    ;Top right cor
          ner
[EN3]   78:                call pos_cur
[78C]   79:                mov al,topright      ;Top righ
          t character
[E02]   80:                int 10h
[758]   81:                call nextline   ;Row 2, column
           80
[LV2]   82:                mov al,vert
[1M2]   83:                push ax
[E02]   84:                int 10h
[C58]   85:                call nextline   ;Row 3, column
           80
[PMB]   86:                mov al,rlink    ;Right linking
          character
[E02]   87:                int 10h
[YWB]   88:                pop ax          ;Restore verti
          cal character
[1M2]   89:                push ax
[UN2]   90:                call loop
```

```
[BK8]   91:                call nextline   ;Row 23, colum
          n 0
[PMB]   92:                mov al,rlink    ;Right linking
           character
[E02]   93:                int 10h
[1M8]   94:                call nextline   ;Row 24, colum
          n 80
[YWB]   95:                pop ax          ;Restore verti
          cal character
[E02]   96:                int 10h
[7M8]   97:                call nextline   ;Row 25, colum
          n 80
[XWC]   98:                mov al,botright      ;Bottom r
          ight character
[E02]   99:                int 10h
[9A8]  100:                mov dx,1701h    ;Line 23, colu
          mn 1
[V5C]  101:                mov si,offset as1   ;adress m
          essage buffer
[VX9]  102:                call dloop      ;Display 'Copy
          right'
[CZ9]  103:                mov dx,0103h    ;Last cursor p
          osition
[EN3]  104:                call pos_cur
[DY2]  105:                mov bx,attr
[VK7]  106:                ret             ;Return to mod
          ule1
[V10]  107: ;
[ZS5]  108: exit:          call cl_com
[GN7]  109:                cmp byte ptr cs:[lind],0
[ZW2]  110:                jz n_step
[6F2]  111:                call get
[YH7]  112: n_step:        mov bh,0        ;Page 1
[EA8]  113:                mov ax,0500h    ;Select page 1
           again
[E02]  114:                int 10h
[126]  115:                mov dx,word ptr[cur]
[3DB]  116:                                ;Restore forme
          r cursor position
[P52]  117:                mov ah,2
[G7A]  118:                int 10h         ;Set cursor in
           page 1 again
[VK7]  119:                ret             ;Return to mod
          ule1
[V10]  120: ;
[834]  121: skip2:         inc dh
[C85]  122: nextline:      inc dh
[XS5]  123: pos_cur:       push ax
[7M2]  124:                push bx
[DM2]  125:                push cx
[WA9]  126:                mov ah,02h      ;Set cursor po
          sition
[E02]  127:                int 10h
[XY1]  128:                pop cx
[SY1]  129:                pop bx
[L52]  130:                pop ax
[EUC]  131:                mov ah,09h      ;Write charact
          er and attribute
[SM0]  132:                ret
[V10]  133: ;
[329]  134: loop:          mov cx,13h      ;Count of rows
[R39]  135: l1:            call nextline   ;Next row
[EN3]  136:                call pos_cur
[DM2]  137:                push cx
[TU8]  138:                mov cl,1        ;Character cou
          nt
[E02]  139:                int 10h
[VN9]  140:                pop cx          ;Restore loop
          counter
[P62]  141:                loop l1
[TU8]  142:                mov cl,1        ;Character cou
          nt
[SM0]  143:                ret
[V10]  144: ;
[866]  145: dloop:         mov bx,attr
[EN3]  146:                call pos_cur
[MX9]  147:                lodsb           ;Load message
          character
[85A]  148:                or al,al        ;Check for ter
          minator 00h
[CM2]  149:                jz enddl
[1X8]  150:                mov cx,1        ;Character cou
          nt
[E02]  151:                int 10h
[E56]  152:                inc dl          ;Next column
[9S2]  153:                jmp dloop
[NV1]  154: enddl:
[SM0]  155:                ret
[V10]  156: ;
[7X2]  157: cur:           dw 0
[V10]  158: ;
[000]  159:
[7M2]  160:                cseg ends
[D13]  161:                end module1
```

**Listing 2. ... und dem zweiten Teil »clkframe.asm«**

```
 81:                cmp al,7          ;Check video mode is
    not graphics
 82:                jnz chk_al
143:                mov bl,70h        ;Attribute
160:                mov bl,0f0h       ;Blinking attribute
184:                cmp al,7
185:                jnz end_int
369:                mov word ptr [dloop+1],000fh  ;Set
     up white on black
553:                mov bx,000fh
561:                mov word ptr[dloop+1],000fh
565:                mov word ptr[dloop+1],000fh
570:                mov word ptr[dloop+1],000fh   ;white
     on black
574:                mov word ptr[dloop+1],000fh   ;white
     on black
580:                mov bx,000fh
```

**Listing 3. Diese 13 Zeilen müssen Sie wie angegeben ändern, um Listing 1 an die Hercules-Grafikkarte anzupassen**

```
 11: attr           equ 0007h         ;White on black
 12: block          equ 0a0bh         ;Block cursor top lin
    e 10. bottom 11
 36:                mov bh,07h        ;Attribute for blanke
    d window
 41:                mov dx,0101h      ;Row 1, column 1
 54:                call nextline
 77:                mov dx,014fh      ;Top right corner
103:                mov dx,0203h      ;Last cursor position
134: loop:          mov cx,12h        ;Count of rows
```

**Listing 4. Auch in Listing 2 müssen Sie für die Hercules-Anpassung einige Zeilen ersetzen**

```
[GN7]     0: E9 45 02 01 00 00 00 00 00 CD AB CD AB 07 00 20
[LK7]     1: 20 20 20 3A 20 20 3A 20 20 20 20 FA 50 1E 9C 0E
[DL7]     2: 1F 53 51 52 56 57 55 06 B8 40 00 8E C0 80 3E 04
[AL7]     3: 01 12 74 07 FE 06 04 01 E9 F8 00 C6 06 04 01 00
[GV7]     4: 80 3E 03 01 00 75 03 E9 C7 00 B4 0F CD 10 3C 03
[ZY7]     5: 7F F5 26 A1 6E 00 B1 04 51 D3 E0 26 8B 1E 6C 00
[VZ7]     6: 8B D3 B1 0C D3 EA 03 D0 8B C3 59 D3 E0 BB 46 44
[AY7]     7: F7 F3 89 16 17 01 BB 3C 00 33 D2 F7 F3 89 16 14
[VH7]     8: 01 E8 2C 00 3B 06 11 01 75 1F A3 11 01 A1 14 01
[1Z7]     9: E8 1D 00 A3 14 01 A1 17 01 F7 E3 BB 46 44 F7 F3
[KX7]    10: E8 0D 00 A3 17 01 EB 10 90 50 E8 51 01 58 EB DA
[1U7]    11: D4 0A 05 30 30 86 E0 C3 B4 03 32 FF CD 10 89 16
[MS7]    12: 05 01 2E 8B 16 07 01 B4 02 CD 10 BE 10 01 B9 0A
[U18]    13: 00 8A 04 56 51 B4 09 B9 01 00 B3 4F CD 10 FE C2
[D58]    14: B4 02 32 FF CD 10 59 5E 46 E2 E6 80 3E 0F 01 0E
[PL7]    15: 75 15 2E 8B 16 07 01 32 FF B4 02 CD 10 B8 0E 09
[M08]    16: B3 CF B9 01 00 CD 10 8B 16 05 01 B4 02 32 FF CD
[AE7]    17: 10 80 3E 0D 01 05 72 18 A1 09 01 3B 06 11 01 75
[UH7]    18: 12 A1 0B 01 3B 06 14 01 75 09 C7 06 0D 01 00 00
[5V7]    19: E8 C0 00 B4 0F CD 10 3C 03 7F 33 2E 80 3E E8 02
[LV7]    20: 00 74 2B E8 34 00 2E 80 3E F2 02 00 74 20 2E FF
[P18]    21: 0E EA 02 75 19 2E C7 06 EA 02 C8 01 2E FE 0E E9
[WT7]    22: 02 75 0B 2E A0 E8 02 2E A2 E9 02 E8 60 00 07 5D
[G38]    23: 5F 5E 5A 59 5B 9D 1F 58 FB CF 2E A1 05 01 2E 3B
[DU7]    24: 06 EC 02 2E A3 EC 02 74 27 BE 63 00 26 8B 14 83
[ZT7]    25: C2 04 BE 65 00 26 8A 04 EE 2E C6 06 F2 02 01 90
[SV7]    26: 2E C7 06 EA 02 C8 01 2E A0 E8 02 2E A2 E9 02 C3
[PV7]    27: 26 A1 17 00 3B 06 EE 02 2E A3 EE 02 75 CB 26 A1
[XY7]    28: 1C 00 3B 06 F0 02 2E A3 F0 02 75 BD EB E1 BE 63
[3X7]    29: 00 26 8B 14 83 C2 04 BE 65 00 26 8A 04 24 F7 EE
[MG7]    30: 2E C6 06 F2 02 00 90 C3 06 02 C8 01 00 00 00 00
[DR7]    31: 00 00 01 C7 06 09 01 CD AB B0 20 A2 0F 01 B0 03
[MV7]    32: E8 1C 00 FE 06 0E 01 B0 04 E8 13 00 FE 06 0E 01
[5P7]    33: B0 05 E8 0A 00 FE 06 0D 01 C6 06 0E 01 00 C3 51
[B08]    34: 50 B0 B6 E6 43 32 C0 E6 42 58 E6 42 E4 61 50 0C
[WB8]    35: 03 E6 61 B9 FF FF B0 02 E2 FE B9 FF 0F FE C8 3C
[M18]    36: 01 74 F5 58 E6 61 59 C3 0E 1F E8 63 06 BE 5B 06
[W08]    37: E8 5A 07 BA 14 0F E8 54 07 BA 14 04 E8 4E 07 BA
[XR7]    38: 14 05 E8 48 07 BA 14 06 E8 42 07 BA 14 07 E8 3C
[WZ7]    39: 07 BA 14 08 E8 36 07 BA 14 09 E8 30 07 BA 14 0A
[718]    40: E8 2A 07 BA 14 0B E8 24 07 BA 14 0C E8 1E 07 BA
[3T7]    41: 14 0D E8 18 07 B8 1C 35 CD 21 26 80 3F CF 74 41
[2F7]    42: E8 65 00 E8 C4 06 CD 20 20 20 20 20 20 20 20 20
[397]    43: 20 20 20 20 20 20 20 20 20 20 20 56 65 72 94 66
[BP7]    44: 66 65 6E 74 6C 69 63 68 74 20 69 6E 20 44 4F 53
[3D7]    45: 20 45 78 74 72 61 20 41 75 73 67 61 62 65 20 34
[G78]    46: 00 BB 00 00 8E C3 8C DB FA BE 72 00 26 89 1C 4E
[Y58]    47: 4E BB 1B 01 26 89 1C FB 8C D8 8E C0 E8 09 00 E8
[C08]    48: 68 06 BA 48 03 42 CD 27 C7 06 AE 0A 0C 01 E8 A9
[5Z7]    49: 01 E8 FF 01 E8 12 02 E8 CD 01 33 C0 33 DB 33 F6
[F28]    50: 32 ED BF 80 00 8A 0D BD 9C 09 80 F9 00 75 03 E9
[5R7]    51: D2 00 47 8A 1D 80 FB 40 7C 13 80 FB 50 7D 06 80
[YW7]    52: CB 60 EB 09 90 80 FB 60 7D 03 80 CB 70 80 FB 65
[HN7]    53: 75 11 26 C6 06 03 01 01 BA 0A 14 BE 80 08 E8 4C
[6Z7]    54: 06 EB A5 80 FB 6C 75 12 26 C7 06 07 01 00 00 BA
[4Y7]    55: 0A 14 BE 9E 08 E8 35 06 EB 8E 80 FB 6D 75 13 26
```

```
[218]    56: C7 06 07 01 23 00 BA 0A 14 BE BC 08 E8 1E 06 E9
[RU7]    57: 76 FF 80 FB 72 75 13 26 C7 06 07 01 46 00 BA 0A
[W68]    58: 14 BE DA 08 E8 06 06 E9 5E FF 80 FB 73 75 12 26
[D08]    59: C6 06 E8 02 06 BA 0A 14 BE F8 08 E8 EF 05 E9 47
[P38]    60: FF 80 FB 62 75 12 26 C6 06 E8 02 00 BA 0A 14 BE
[EZ7]    61: 16 09 E8 D8 05 E9 30 FF 80 FB 61 75 12 26 C6 06
[Y08]    62: 03 01 00 BA 0A 14 BE 26 08 E8 C1 05 E9 19 FF 80
[UW7]    63: FB 7A 75 13 26 C7 06 09 01 CD AB 26 C6 06 0F 01
[108]    64: 20 E9 04 FF E9 B2 00 80 FB 20 75 58 E2 09 81 FD
[S58]    65: 9C 09 75 06 E9 F1 FE E9 18 FF 81 FD A1 09 75 14
[9X7]    66: A1 9D 09 86 E0 3D 33 32 7F 0A A1 A0 09 86 E0 3D
[T78]    67: 39 35 7E 0C BE 34 09 BA 0A 14 E8 70 05 E9 C8 FE
[AW7]    68: 8B 2E 9D 09 26 89 2E 09 01 8B 2E A0 09 26 89 2E
[XU7]    69: 0B 01 26 C6 06 0F 01 0E BE 44 08 BA 0A 14 E8 4C
[368]    70: 05 E9 A4 FE 80 FB 3A 74 05 80 FB 2E 75 2E 81 FD
[0T7]    71: 9E 09 75 08 83 F9 03 72 BB 45 EB 90 81 FD 9D 09
[AU7]    72: 75 B2 53 8A 1E 9D 09 88 1E 9E 09 C6 06 9D 09 30
[MW7]    73: 5B 83 F9 02 72 9E 83 C5 02 E9 70 FF 80 FB 30 72
[828]    74: 0C 80 FB 39 77 07 45 88 5E 00 E9 5F FF BE 52 09
[Z68]    75: BA 0A 14 E8 F7 04 E9 4F FE C3 26 A1 09 01 3D CD
[TN7]    76: AB 74 1A C7 06 9F 09 3A 00 A3 9D 09 26 A1 0B 01
[738]    77: A3 A0 09 BA 0A 12 BE 8E 09 E8 D1 04 C3 BA 0A 12
[108]    78: BE 62 08 E8 C7 04 C3 BB 17 01 BA 20 0F E8 9C 04
[208]    79: BD 7F 00 C6 46 00 07 8B D5 B4 0A CD 21 C7 06 AE
[N88]    80: 0A 1F 01 BA 0A 14 BE 3D 06 E8 A1 04 C7 06 AE 0A
[458]    81: 0C 01 C3 C7 06 AE 0A 1F 01 BA 20 0F BE 47 06 E8
[L28]    82: 8B 04 C7 06 AE 0A 0C 01 C3 BA 00 00 BB 0C 00 E8
[AF7]    83: 5A 04 B9 50 00 B8 20 09 CD 10 C3 20 00 20 20 20
[G37]    84: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
[237]    85: 20 20 20 20 20 20 20 20 20 20 00 20 20 20 20 20
[G97]    86: 20 20 20 20 20 20 20 20 20 20 20 43 6C 6F 63
[7U7]    87: 6B 2D 55 74 69 6C 69 74 79 20 49 49 20 76 6F 6E
[8R7]    88: 20 44 4F 53 20 49 6E 74 65 72 6E 61 74 69 6F 6E
[7F7]    89: 61 6C 00 49 68 72 65 20 57 61 68 6C 20 2D 20 00
[GF7]    90: 45 28 69 6E 29 20 20 20 20 2D 20 41 6E 7A 65
[SP7]    91: 69 67 65 20 65 69 6E 73 63 68 61 6C 74 65 6E 20
[S77]    92: 20 20 20 20 20 20 00 41 28 75 73 29 20 20 20 20
[XN7]    93: 20 2D 20 41 6E 7A 65 69 67 65 20 61 75 73 73 63
[WC7]    94: 68 61 6C 74 65 6E 20 20 20 20 20 20 20 00 4C 28
[YP7]    95: 69 6E 6B 73 29 20 20 20 2D 20 41 6E 7A 65 69 67
[AK7]    96: 65 20 6F 62 65 6E 20 6C 69 6E 6B 73 20 20 20 20
[9F7]    97: 20 20 20 20 00 4D 28 69 74 74 65 29 20 20 20 2D
[KW7]    98: 20 41 6E 7A 65 69 67 65 20 6F 62 65 6E 20 69 6E
[0F7]    99: 20 64 65 72 20 4D 69 74 74 65 20 00 52 28 65 63
[2L7]   100: 68 74 73 29 20 20 2D 20 41 6E 7A 65 69 67 65 20
[BE7]   101: 6F 62 65 6E 20 72 65 63 68 74 73 20 20 20 20 20
[YE7]   102: 20 20 00 48 48 3A 4D 4D 20 20 20 20 20 2D 20 41
[1S7]   103: 6C 61 72 6D 7A 65 69 74 20 61 75 66 20 48 48 3A
[CN7]   104: 4D 4D 20 73 65 74 7A 65 6E 00 5A 28 75 72 81 63
[AR7]   105: 6B 29 20 20 2D 20 41 6C 61 72 6D 20 7A 75 72 81
[BH7]   106: 63 6B 73 65 74 7A 65 6E 20 20 20 20 20 20 20 20
[SL7]   107: 00 53 28 74 61 72 74 29 20 20 20 2D 20 4D 6F 6E
[1M7]   108: 69 74 6F 72 73 63 68 75 74 7A 20 73 74 61 72 74
[DN7]   109: 65 6E 20 20 20 20 20 00 42 28 65 65 6E 64 65 6E
[XS7]   110: 29 20 2D 20 4D 6F 6E 69 74 6F 72 73 63 68 75 74
[0H7]   111: 7A 20 62 65 65 6E 64 65 6E 20 20 20 20 20 00 5B
[3T7]   112: 45 6E 74 65 72 5D 20 20 20 2D 20 43 6C 6F 63 6B
[1U7]   113: 2D 55 74 69 6C 69 74 79 20 76 65 72 6C 61 73 73
[8W7]   114: 65 6E 20 20 20 00 43 6C 6F 63 6B 2D 41 6E 7A
[BM7]   115: 65 69 67 65 20 61 75 73 67 65 73 63 68 61 6C 74
[2M7]   116: 65 74 20 00 20 4E 65 75 65 20 41 6C 61 72 6D 7A
[5R7]   117: 65 69 74 20 65 69 6E 67 65 73 74 65 6C 6C 74 20
[LL7]   118: 20 00 20 4B 65 69 6E 65 20 41 6C 61 72 6D 7A 65
[NM7]   119: 69 74 20 65 69 6E 67 65 73 74 65 6C 6C 74 20 00
[WP7]   120: 20 43 6C 6F 63 6B 2D 41 6E 7A 65 69 67 65 20 65
[2K7]   121: 69 6E 67 65 73 63 68 61 6C 74 65 74 20 00 20 41
[D08]   122: 6E 7A 65 69 67 65 20 6F 62 65 6E 20 6C 69 6E 6B
[DB7]   123: 73 20 20 20 20 20 20 20 20 20 20 00 20 41 6E 7A
[MH7]   124: 65 69 67 65 20 6F 62 65 6E 20 69 6E 20 64 65 72
[XK7]   125: 20 4D 69 74 74 65 20 20 20 00 20 41 6E 7A 65 69
[WG7]   126: 67 65 20 6F 62 65 6E 20 72 65 63 68 74 73 20 20
[HG7]   127: 20 20 20 20 20 20 00 20 4D 6F 6E 69 74 6F 72
[2N7]   128: 73 63 68 75 74 7A 20 69 73 74 20 61 6B 74 69 76
[5P7]   129: 20 20 20 20 20 00 20 4D 6F 6E 69 74 6F 72 73 63
[FP7]   130: 68 75 74 7A 20 69 73 74 20 69 6E 61 6B 74 69 76
[YF7]   131: 20 20 20 00 20 55 68 72 7A 65 69 74 20 66 61 6C
[EH7]   132: 73 63 68 20 64 65 66 69 6E 69 65 72 74 20 21 20
[YF7]   133: 20 00 20 50 61 72 61 6D 65 74 65 72 20 69 73 74
[LG7]   134: 20 75 6E 62 65 6B 61 6E 6E 74 20 21 20 20 20 00
[G37]   135: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
[347]   136: 20 20 20 20 20 20 20 20 20 20 20 20 00 20 41
[RR7]   137: 6C 61 72 6D 7A 65 69 74 20 69 73 74 20 20 20 3A
[G27]   138: 20 20 20 20 20 20 20 20 20 20 00 00 00 00 00
[RX7]   139: B7 00 B4 03 CD 10 89 16 C2 0A B8 01 05 CD 10 B9
[UU7]   140: 07 06 B4 01 CD 10 B8 00 06 B7 1F 33 C9 BA 50 18
[EY7]   141: CD 10 BB 1F 01 BA 01 00 E8 B1 00 B1 4E B0 CD CD
[1V7]   142: 10 E8 A4 00 CD 10 B6 16 E8 A1 00 CD 10 E8 98 00
[DT7]   143: CD 10 33 D2 E8 95 00 B1 01 B0 C9 CD 10 E8 8A 00
[NX7]   144: B0 BA 50 CD 10 E8 82 00 B0 CC CD 10 58 50 E8 88
[F08]   145: 00 E8 76 00 B0 CC CD 10 E8 6F 00 58 CD 10 E8 69
[2X7]   146: 00 B0 C8 CD 10 BA 4F 00 E8 61 00 B0 BB CD 10 E8
[PX7]   147: 58 00 B0 BA 50 CD 10 E8 50 00 B0 B9 CD 10 58 50
[HU7]   148: E8 56 00 E8 44 00 B0 B9 CD 10 E8 3D 00 58 CD 10
[NW7]   149: E8 37 00 B0 BC CD 10 BA 01 17 BE A8 03 E8 4D 00
[158]   150: BA 03 01 E8 26 00 BB 1F 01 C3 E8 A6 FB 2E 80 3E
[9T7]   151: 80 00 00 74 03 E8 6F FB B7 00 B8 00 05 CD 10 8B
[GV7]   152: 16 C2 0A B4 02 CD 10 C3 FE C6 FE C6 50 53 51 B4
[248]   153: 02 CD 10 59 5B 58 B4 09 C3 B9 13 00 E8 EB FF E8
[2X7]   154: EA FF 51 B1 01 CD 10 59 E2 F2 B1 01 C3 BB 1F 01
[328]   155: E8 D9 FF AC 0A C0 74 09 B9 01 00 CD 10 FE C2 EB
[PK1]   156: EC C3 00 00
```

**Listing 5. Wenn Sie das Clock-Utility in der übersetzten Form als Hexdump eingeben, sparen Sie viel Zeit beim Abtippen**

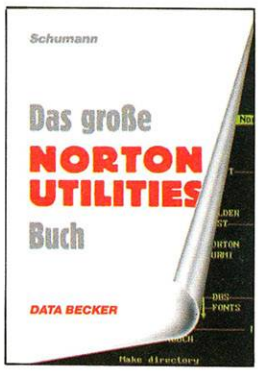
# Zum Beispiel: DATA WELT.

Die DATA WELT – jetzt völlig neu. Im neuen Layout. Mit neuem Inhalt.
Die neue DATA WELT konzentriert sich nun voll und ganz auf die MS-DOS-Rechner.
Geblieben ist, was die DATA WELT so erfolgreich gemacht hat: schonungslose Produkt-Tests, aktuelle News, kompetente Kaufhilfen und viele Tips & Tricks. Ausschließlich für die MS-DOS-WELT.
Aktuell informiert sein, heißt regelmäßig DATA WELT lesen.
Monat für Monat.
Die neue DATA WELT – jetzt überall im Zeitschriftenhandel.

# AKTUELL!

## Zum Beispiel: Norton Utilities.

uch ohne Vorkenntnisse die Norton-
ilities sinnvoll einsetzen? Nichts
ichter als das! Mit dem Norton-
ilities-Buch. Anhand einer Übungs-
iskette, die
ie selbst
rstellen, zeigt
nnen dieses
uch, was alles
n dem Nor-
on-Utilities-
aket steckt –
eispielsweise
n Sachen
atenschutz.
owohl in den
ersionen 3
nd 4 als auch
n der Advan-
ed Edition.
abei stoßen
ie nicht nur
uf eine Reihe
eißer Tips,
ondern Sie erfahren auch noch jede
lenge über die internen Vorgänge
hres Rechners.




**Das Norton-Utilities-Buch**
**Hardcover, inkl. Diskette**
**ca. 300 Seiten, DM 49,–**
**erscheint ca. 9/88**

## Zum Beispiel: MS-Windows

as große Windows-Buch sorgt für
lare Sicht bei der Arbeit mit dieser
berfläche. Hier wird die ganze
alette der Windows-Hilfsprogramme
nd Windows-
pplikationen
usführlich
eschrieben.
atürlich
rfahren Sie
uch, wie Sie
it MS-DOS
nter Win-
ows arbeiten,
ie sich mit
er WIN.INI-
atei die Welt
on Windows
erändern läßt
nd was Sie
ei der Instal-
tion von
indows auf
erschiedenen

**Das große Windows-Buch**
**Hardcover,**
**ca. 350 Seiten, DM 49,–**
**erscheint ca. 8/88**

Rechnern beachten müssen. Besonderer Leckerbissen nicht nur für Kreative: DTP mit Windows. Detailliert beschreiben die Autoren, wie Sie mit Windows-Write und -Paint ansprechende professionelle Ergebnisse erzielen. Selbstverständlich werden dabei die verschiedenen Windows-Versionen (1.XX, 2.XX, 386) berücksichtigt.

## Zum Beispiel: QuickBASIC.

Das große PC-BASIC-Buch – unentbehrlich für jeden ernsthaften BASIC-Programmierer. Ein paar Stichworte aus dem Inhalt zeigen, warum: Variablen und Datentypen, Zeichenkettenverarbeitung, mathematische Funktionen und Berechnungen, Grafik und Sound, Erstellen und Einbinden von Assembler-Routinen, professioneller Umgang mit dem Editor, Fehlerbehandlung, Interrupt-Programmierung, Dateiverwaltung, Druckerausgabe und und und. Ob PC-, GW-, Quick- oder Turbo-BASIC – hier werden Sie umfassend informiert und bekommen eine Übersicht über die ungeheuren Möglichkeiten professioneller PC-BASIC-Programme. Dazu noch eine ganze Reihe fertiger Utilities, mit denen Sie Ihre Arbeit am PC noch effektiver gestalten können.

**Das große PC-BASIC-Buch**
**Hardcover, inkl. Diskette,**
**ca. 400 Seiten, DM 69,–**
**erscheint ca. 9/88**

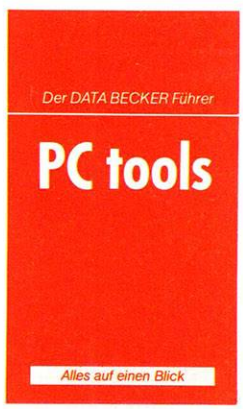
## Zum Beispiel: MS-Works.

MS-Works bietet alles für die tägliche Arbeit am PC: Textverarbeitung, Tabellenkalkulation, Geschäftsgrafik, Datenbank und Kommunikation. Mit MS-Works für Einsteiger haben Sie dieses komplexe Programmpaket schnell im Griff. Von Anfang an. Ob Installation, Arbeit mit den einzelnen Modulen, Datenaustausch zwischen den MS-Works-Anwendungen oder die Kommunikation mit einem anderen PC – Sie wissen immer, worauf es ankommt. MS-Works für Einsteiger – leicht verständlich und mit vielen praktischen Beispielen.

**MS-Works für Einsteiger**
**256 Seiten, DM 29,–**

## Zum Beispiel: PC-Tools.

PC-Tools ist wohl eines der verbreitetsten Hilfsprogramme für den PC. Mit dem DATA BECKER Führer haben Sie dieses mächtige Werkzeug jederzeit sicher im Griff. Alle Funktionen von PC-Tools werden in alphabetischer Reihenfolge kurz aber verständlich erklärt. Selbstverständlich finden Sie hier auch eine Gegenüberstellung der Version 1, 2, 3 und 4. Ebenfalls im Buch: eine grundsätzliche Erläuterung des Befehlssatzes. Wann immer Sie mal nicht weiterwissen, ein Blick in den DATA BECKER Führer zu PC-Tools genügt.




**Der DATA BECKER Führer**
**PC Tools**
**ca. 250 Seiten, DM 29,80**
**erscheint ca. 9/88**

## COUPON

HIERMIT BESTELLE ICH

NAME, VORNAME

STRASSE

ORT

zzgl. DM 5,– Versandkosten unabhängig von der bestellten Stückzahl
☐ per Nachnahme ☐ Verrechnungsscheck liegt bei

**DATA BECKER**
erowingerstr. 30 · 4000 Düsseldorf · Tel. (0211) 310010

# Befreien Sie Ihren Speicher

**Es ist Ihnen sicher auch schon einmal widerfahren, daß MS-DOS Ihnen beim Laden eines größeren Programmes mitteilte: »not enough memory«. Speicherresidente Utilities haben den Hauptspeicher bis zum Rand gefüllt. »Wie kann ich diese Programme wieder loswerden?«, lautet die Frage. Das Dilemma endet dann meist mit der Tastenkombination [Ctr-Alt-Del]. Soweit müssen Sie es aber nicht kommen lassen, wenn Sie mit den Programmen »nodi« und »clean« arbeiten.**

So nützlich speicherresidente Programme oft sind , um so mehr stören sie beim Versuch, mit einem großen Programmpaket, beispielsweise mit Grafikanwendungen, zu arbeiten. Ganz abgesehen davon, daß manch einer auch schon vergessen hat, was seine Shift-, Control- und Alt-Tasten überhaupt noch bewirken. Das zweite Problem bei der Arbeit mit speicherresidenten Utilities ist, daß sich einige von Ihnen nicht miteinander vertragen. Das ist der Fall, wenn Programm A den Hardware-Tastatur-Interrupt »Int 9hex« auf den unbenutzten »Interrupt 62hex« legt, um selbst mit dem neu installierten Interrupt etwaige »Hotkeys« abzufragen. Die normale Arbeit mit der Tastatur wird in dieser neuen Routine durch den Aufruf der alten (Interrupt 62hex) gewährleistet. Soweit so gut. Programm B macht nun dasselbe, das bedeutet, es legt auch eine alte Interrupt-Routine auf den mittlerweile jedoch vergebenen Interrupt 62hex. Beim nächsten Tastendruck stürzt der Rechner unweigerlich ab, selbst die drei Tasten [Ctrl-Alt-Del] werden von der Tastatur nicht mehr akzeptiert. Sind Sie nicht im Besitz einer Resettaste, dann heißt das für Sie: Rechner ausschalten, kurz abwarten und danach wieder neu starten. Dies ist eine besonders lästige Prozedur, weil der Rechner nach dem Einschalten routinemäßig seine Hardware überprüft und erst daraufhin das System neu bootet. Außerdem hat sich dabei schon so manche Festplatte für immer von ihrem Besitzer verabschiedet.

Doch selbst speicherresidente Programme, die sich durch erneuten Aufruf wieder entfernen lassen, haben ihre Haken. Das Programm ist zwar meist vollständig entfernt, aber MS-DOS zerstört unsere Illusion vom »befreiten RAM« sehr schnell wieder. Hauptverantwortlich dafür ist die interne Speicherverwaltung des Betriebssystems. Dazu ein Beispiel:

Sie laden eine MS-DOS-Benutzeroberfläche, wie Quick-DOS oder den Norton-Commander, und das Programm verbleibt mit zirka 60 KByte resident im Hauptspeicher. Von dieser Oberfläche aus starten Sie Sidekick, das sich selbst mit weiteren 40 KByte resident im RAM verankert. Kehren Sie von der erstgenannten Benutzeroberfläche zum normalen MS-DOS zurück, werden die 60 KByte zwar anstandslos wieder zur Verfügung gestellt, doch haben sie keinen Anschluß mehr an den freien Speicherbereich, der erst nach Sidekick beginnt. Das so entstandene »Loch« kann nur noch von Programmen mit der Extension »exe« genutzt werden, die kleiner sind als 60 KByte. Programmen mit der Endung »com« wird nämlich standardmäßig der gesamte noch freie Hauptspeicher zur Verfügung gestellt, da MS-DOS nicht weiß, wieviel Platz sie benötigen. Dieser beginnt in unserem konkreten Beispiel erst nach Sidekick. Aber auch größere Exe-Dateien, wie Grafik- oder Kalkulationsprogramme können mit solchen Lücken im Speicher nichts anfangen, da der interne »memory manager« von MS-DOS eine Aufteilung von Programmen im RAM nicht unterstützt.

Dieser Mangel des Betriebssystems läßt sich zwar nicht beheben, doch stellt Ihnen MS-DOS einige Funktionen zur Verfügung, die zumindest eine Beeinflussung der Speicherverwaltung erlauben. Einen derartigen Lösungsweg stellt die MS-DOS-Funktion 58hex dar. (Bild 1) Diese Funktion gestattet mit ihrer ersten Unterfunktion einen Einblick in die derzeitige Strategie der Speicherverteilung. Unterfunktion zwei erlaubt Ihnen selbst, diese Strategie zu bestimmen. Zur Auswahl stehen insgesamt drei Arten der Speicherverwaltung, von denen eine eine Besonderheit aufweist.

Die erste Strategie kann man als »Suche von unten« bezeichnen. Sie veranlaßt MS-DOS, den RAM – beginnend bei den hohen, hin zu den niedrigen Adressen – zu durchsuchen und den ersten passenden, das heißt ausreichend großen Speicherbereich, dem anfordernden Programm zur Verfügung zu stellen. Die zweite Alternative, die »Suche von oben«, funktioniert bis auf die geänderte Richtung ebenso wie die eben vorgestellte Suche von unten. Eine besonders effiziente Methode stellt MS-DOS mit der »Suche nach dem besten Platz« zur Verfügung. Bei dieser dritten Strategie versucht MS-DOS dem anfordernden Programm einen Speicherbereich zur Verfügung zu stellen, der, wenn möglich, nicht viel größer ist als der mindestens benötigte. Der positive Effekt hierbei liegt darin, daß nur wenig Speicherplatz verschwendet wird. Wie bereits erwähnt, profitieren davon aber nur Exe-Dateien, die MS-DOS mitteilen, wieviel Mindestplatz sie im Speicher benötigen.

Auf das konkrete Problem übertragen, würde sich dies folgendermaßen auswirken: Sowohl Quick-DOS als auch Sidekick müßten entfernt werden, um Sidekick danach platzsparend zu installieren. Doch wie entfernt man Sidekick?

## Sidekick läßt sich nicht aus dem RAM verbannen. Mit »clean.exe« schaffen Sie Abhilfe

Um speicherresidente Programme aus dem RAM zu löschen, müssen mehrere Ziele verfolgt werden. Zum einen ist das residente Programm erst einmal aufzuspüren, damit Sie es später entfernen können, und zum anderen müssen Sie die von ihm veränderten Parameter wieder restaurieren. Diese letzte Aufgabe erweist sich als nicht sehr schwierig, da die wichtigste Veränderung, die ein residentes Utility am System vornimmt, die Adressen der Interrupts betrifft. Diese Adressen, auch als Interruptvektoren bezeichnet, stehen in einem sogenannten »Interruptvectortable« am Anfang des Hauptspeichers (»0000:0000« bis »0000:0400hex«).

Um dieses 1024 Byte große »Adressenfeld« vor dem Laden eines residenten Programmes zu konservieren, gibt es zwei Alternativen. Entweder legt man die »Orginaladressen« in einer Disketten-Datei ab, oder man konserviert die

| Unterfunktion | Eingabe | Ausgabe |
|---|---|---|
| 0 : Get memory management | AX = 5800hex | AX = Art der Verwaltung<br>0: Suche von unten<br>1: Suche besten Platz<br>2: Suche von oben |
| 1 : Set memory management | AX = 5801hex<br>BX = Art der Verwaltung | evtl. Carry-Flag = 1<br>d.h. Fehler aufgetreten |

**Bild 1. Diagramm der Unterfunktionen der MS-DOS-Funktion »58hex«**

Adressen im Datenbereich eines (residenten) Programms. Die erste Möglichkeit ist wegen der Unhandlichkeit und Unsicherheit des Verfahrens unrentabel, zumal die Alternative, das Sichern der Systemparameter in einem residenten Programm nicht nur den Vorteil der Übersichtlichkeit aufweist, sondern auch noch eine Marke im Hauptspeicher darstellt, bis zu der der RAM-Speicher später wieder gesäubert werden kann. Das »Einfrieren« des Interruptvectortable besorgt das kleine Programm »nodi.com«. In einer Schleife (»hier«) werden die Bytes von »0000:0000« bis »0000:0400hex« in den 1024 Byte großen Datenbereich am Anfang des Programmes kopiert. Danach veranlaßt »nodi.com« mit den Zeilen

```
mov dx,offset anfang+1
int 27h
```

das Verbleiben der Daten im RAM, der eigentliche Programmcode ab »anfang« hingegen wird von MS-DOS wieder überschrieben. Dank dieser Technik fällt »nodi« mit seinen 1296 Byte, die es im RAM verbraucht, kaum ins Gewicht, und Sie können es ohne schlechtes Gewissen mehrmals laden, um mehrere Marken zu setzen.
In Listing 1 sehen Sie den Assembler-Quelltext »nodi.asm«. Das lauffähige Programm erhalten Sie durch folgende Anweisungen:

```
masm nodi.asm
link nodi.obj
Exe2bin nodi.exe nodi.com
del nodi.exe
```

Einfacher ist die Sache jedoch mit dem MS-DOS-Debugger und der Datei »nodi.deb« (Listing 3). Durch den Befehl

```
debug<nodi.deb
```

befindet sich jetzt die Datei »nodi.com« im aktuellen Verzeichnis auf Ihrer Diskette oder Festplatte. Wie oben bereits angesprochen, ist es durchaus sinnvoll, mehrere Marken durch mehrmaliges Installieren von »nodi.com« zu setzen. Eine Einbindung in die Datei »autoexec.bat« empfiehlt sich auf jeden Fall. Fügen Sie »nodi« aber erst nach dem Eintrag »keybgr« in die Batch-Datei ein, da Sie sonst Gefahr laufen, den deutschen Tastaturtreiber zu entfernen. Dann wären die Tasten [y] und [z] auf Ihrer Tastatur vertauscht.

## Auch Quick-DOS läßt sich entfernen

Das nächste Problem, das Aufspüren der residenten Utilities, erweist sich durch die schlechte MS-DOS-Dokumentation als außerordentlich schwierig. Eine undokumentierte MS-DOS-Funktion ist des Rätsels Lösung. Es handelt sich um die Funktion 52hex, die in ES:BX einen Pointer auf eine MS-DOS-interne Infotabelle übergibt. Diese inoffizielle Tabelle birgt an ihrem »Offset-4« einen Pointer auf den ersten »Memory Control Block« (kurz MCB, das Herz der MS-DOS-internen Speicherverwaltung). Da der Offsetteil dieser Adresse immer Null beträgt, reicht es aus, die an »Offset-2« stehende Segmentadresse zu benutzen. Vor der Darstellung der inneren Struktur des MCB soll zunächst auf die grundlegende Organisation des MCB eingegangen werden.
Die Organisation des MCB erinnert beinahe an eine »Schnitzeljagd« bei der man an jeder Station erfährt, wo es weitergeht (Bild 2). Das erste MCB gibt allerdings keine Auskunft darüber, wo sich das darauffolgende MCB genau befindet. Es übergibt also keine Adresse, sondern teilt nur mit, wie lang es selbst ist. Diese Information erlaubt dann die Ermittlung des nächsten MCB-Anfangs. Die dazu notwendige Formel sieht in Listing 3 folgendermaßen aus:

```
newmcb_adr:=new_mcbadr+lang+1;
```

Wobei die Variable »lang« die Anzahl der Paragraphen enthält, die dieses MCB überwacht. Ob überhaupt noch ein weiteres MCB vorhanden ist, zeigt die MCB-interne Kennung »M« oder »Z« an, wobei die Kennung »M« aussagt, daß noch weitere MCBs im RAM folgen.

## Am einfachsten gestaltet sich der Einbau in die »autoexec.bat«

Wenn Sie das Programm »clean.exe« laden und somit diese Verkettung dechiffrieren, steht es, da es als letztes in das RAM geladen wurde, unter dem letzten MCB mit der Kennung »M«. Da es nicht resident im Speicher verankert wird, wird es von MS-DOS bei der nächsten Gelegenheit vom nächsten Programm überschrieben. Nur wenn sich das Programm resident macht, erweitert sich die MCB-Kette um ein Glied, und das nächste Programm wird in den Speicher zu dem mit »Z« (jetzt noch »Z«, nach dem Laden dann »M«) gekennzeichneten MCB geladen. Das hört sich zwar etwas kompliziert an, erklärt sich aber bei der Verwendung von »clean.exe« von selbst.

Außer dem Platz für das eigentliche Programm richtet MS-DOS zusätzlich noch ein Feld ein, das als Environment (Programmumgebung) bezeichnet wird. Dieses Environment enthält einige für das System unbedingt erforderliche Informationen, die MS-DOS dort eingetragen hat. Die wichtigsten Angaben beziehen sich auf den aktuellen Suchpfad des Kommandoprozessors (»command.com«), den »prompt« und den Namen des zu dem Environment gehörenden Programms.

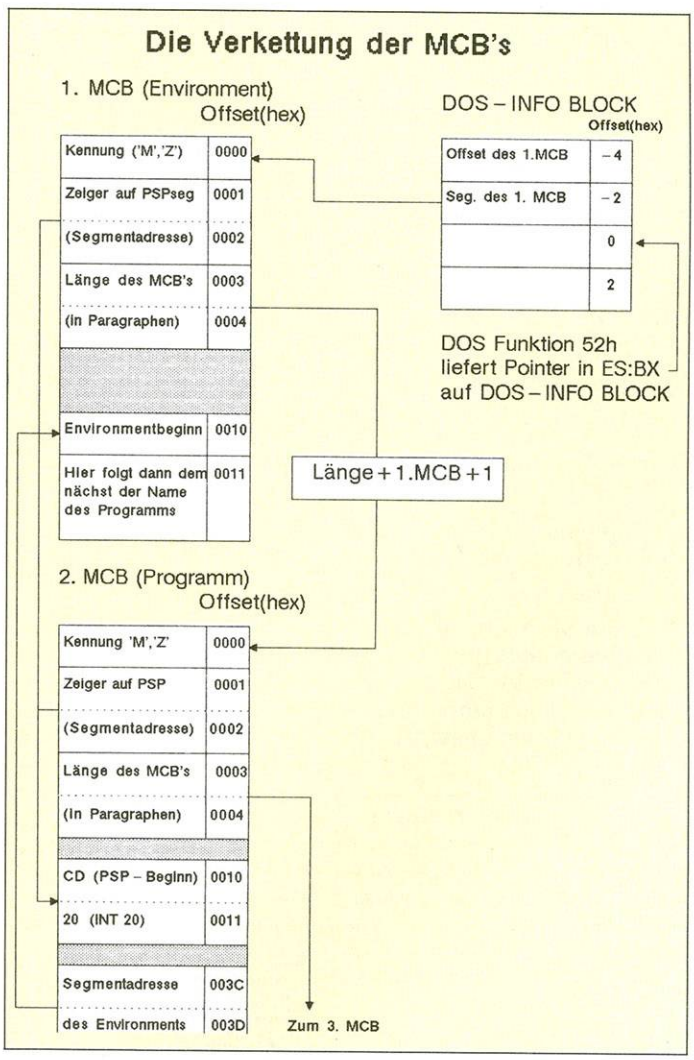



**Bild 2. Diagramm der Verkettung der Memory Control Blöcke im RAM**

Dank seines »program segment prefix« weiß jedes Programm, wo sich sein Environment befindet. Dieser »program segment prefix«, kurz PSP, wird von MS-DOS jedem Programm vorangestellt, das zur Ausführung kommt. Die ersten beiden Bytes im PSP sind immer »CD 20«, was disassembliert »int 20« heißt. Dieser Interrupt 20hex, ein Relikt aus den frühen MS-DOS-Versionen, diente dem Beenden eines Programmes und der Rückkehr zum Aufrufenden. Für Sie sind diese beiden Bytes am Anfang des PSP eine gute Hilfe, um Programme eindeutig zu identifizieren.

```
if memw[mcbseg:$10]=$CD20 then
begin
prog:=true;
```

Aber noch einmal zurück zu dem Environment, das MS-DOS für jedes Programm einrichtet. Im eben vorgestellten PSP befindet sich an dem »Offset 2Chex« die Segmentadresse des Environments. Wichtig ist die Tatsache, daß sowohl Programme als auch Environments direkt auf ihr eigenes MCB folgen. Das heißt, dem MCB an Speicherstelle »3410:0000« folgt das Programm oder Environment an Stelle 3410:0010, das bedeutet das Versetzen um einen Paragraphen (16 Byte) weiter, was gleichbedeutend ist mit der Adresse »3411:0000«.

Durch die interne Umrechnung der Segment- und Offsetadresse in eine 20-Bit-Zahl sollten Sie sich nicht verunsichern lassen. Diese physikalische Adresse, wie die 20-Bit-Adresse auch genannt wird, entsteht ganz einfach: Segmentadresse multipliziert mit 16 (das heißt, dem Hexadezimalwert wird eine Null angehängt) plus Offsetadresse. Die Adresse »3410:0010« ist also tatsächlich mit der Adresse »3411:0000« identisch. Dieser 256 Byte lange Vorspann übermittelt dem Programm wichtige Informationen zur Laufzeit. So zum Beispiel, wohin beim Auftreten eines Fehlers oder bei einem Programmstop durch [Control-Break] verzweigt werden soll. Aber nicht nur unvorhergesehene Ereignisse bedienen sich des Adreßfeldes im PSP, sondern auch bei einem normalen Programmende wird zu einer im PSP abgelegten Adresse gesprungen. Diese letzte sogenannte »Terminate adress« ist auch im Interruptvectortable als Adresse des Interrupt 22hex, des »Terminate-vector« gespeichert. Diese 4-Byte-Adresse spielt nachher beim vollständigen Restaurieren des Systems noch eine wichtige Rolle. Diese vier Bytes werden übrigens, wie alle anderen Adressen auch, nach Intelkonvention »verdreht« im PSP abgelegt.

Mit »debug« erhalten Sie durch folgenden Befehl die aktuelle Adresse des Interrupt 22hex:

```
d 0000:0088 l4
```

Erhalten Sie als Ergebnis etwa die Bytefolge »12 45 65 44«, so bedeutet dies die Adresse »4465:4512« in Segment- und Offset- Schreibweise.
Wie die MCBs untereinander zusammenhängen, haben Sie schon kennengelernt. Wie aber sieht ihre innere Struktur aus? Jedes MCB besteht aus insgesamt fünf Bytes. Das erste Byte am »Offset 0« stellt die schon vorgestellte Kennung des MCB dar, es ist entweder auf 4Dhex (»M«) oder auf 5Ahex (»Z«) gesetzt. Byte zwei und drei, beginnend an »Offset 1« stellen einen Pointer auf den Besitzer des MCB dar, der sowohl ein eigentliches Programm als auch ein Environment sein kann. Bei einem Programm zeigt der Pointer direkt auf das folgende PSP bei einem Environment auf das PSP des Besitzerprogramms. Byte vier und fünf, ab »Offset 3« schließlich geben Auskunft über die Länge des MCB (in Paragraphen) und somit über den Beginn des nächsten MCB.

| Recordteil | Inhalt | woher | wie lang |
|---|---|---|---|
| 1. Kennung | 'M' oder 'Z' | MCBseg : 0000 | char (1 Byte) |
| 2. Cler | Adresse ab der, der Speicher später wieder freigegeben wird | MCBseg + 1 | word (2 Byte) |
| 3. Segment | Segmentadresse des MCB (MCBseg) | MCBseg | " |
| 4. Owner | Segmentadresse des zugehörigen PSP | MCBseg : 0001 | " |
| 5. Länge | Länge des Speicherblocks, der von diesem MCB kontrolliert wird | MCBseg : 0003 (in Paragraphen ! 1 Paragraph = 16 Bytes) | longint (4 Byte) |
| 6. Prog | wenn das MCB ein Programm kontrolliert und kein Environment dann : true | Wenn MCBseg : 0010hex = CD 20 (INT 20) dann prog=true | boolean (1 Byte) |
| 7. Env | (wenn prog true) Segmentadresse des zugehörigen Environments | MCBseg : 003Chex (=PSPseg : 002Chex) | word (2 Byte) |
| 8. Langenv | (wenn prog true) Länge des Environments | Env-1 : 0003 (Env-1=MCBseg des zugehörigen Programmes s. 4.) | " |

**Bild 3. Diagramm der Struktur des Record »ausg«**

## Am wichtigsten ist das Auslesen der MCB-Informationen

Die wichtigste Aufgabe des in Listing 2 vorgestellten Programms »clean.exe« ist das Auslesen der MCB-Informationen.
Die zentralen »procedures« dabei sind »get« und »datenholen«, die alle wichtigen Daten der residenten Programme in den »record ausg« einlesen (Bild 3). Dabei sucht »get« über die MS-DOS-Funktion 52hex den Einstieg in die MCB-Kette, und »new_mcbadr« ermittelt die darauf folgenden Einträge mit Hilfe der von »get« übergebenen Adresse. Der Name des residenten Programms wird im letzten Teil des »record ausg«, in dem Feld »identy« gespeichert. Um diesen Namen zu erhalten, muß »clean« das Environment der residenten Programme nach den Bytekombinationen »00« und »01 00« absuchen. Dabei kennzeichnet »01 00« den Pfadanfang und »00« das Ende des Namens beziehungsweise des Pfades. Die Abfrage, ob es sich bei dem Namen um »nodi.com« handelt, dient der Identifikation der Marke, bis zu der man den Hauptspeicher freigeben kann. Das aktuelle Segment von »nodi« im RAM wird dabei in die Variable »segnodi«, die laufende Nummer in »nodicount« abgelegt.
Um vor gefährlichen Namensverwechslungen seitens des Betriebssystems gefeit zu sein, werden auch die ersten drei charakteristischen Bytes von »nodi« abgefragt:

```
if (memw[ausg[counter].owner:$100]=$00E9)
and (mem[ausg[counter].owner:$102]=$04) then
begin
```

Diese Bytekombination ist nichts anderes als der erste Befehl von »nodi« in Listing 1:

```
jmp anfang
```

Die angesprochene Namensvertauschung kommt dadurch zustande, daß Programme, die sich ihres eigenen Environments entledigen, von MS-DOS das Environment und somit auch den Namen des im Speicher folgenden Programms zugeteilt bekommen. Die eigentliche Speicherfreigabe läuft in zwei Schritten ab. Zuerst wird der Interruptvectortable durch die, in »nodi« ab Offset »103hex« stehenden »alten« Interruptvektoren ersetzt und anschließend erst der Speicher freigegeben. Durch diese Reihenfolge kann ein Interruptvektor niemals ins Leere zeigen. Die Freigabe des Speichers besorgt für Sie die MS-DOS-Funktion 49hex, da nicht das Segment des freizugebenden MCB, sondern das Segment des Besitzers (»cler«) in »ES« übergeben werden muß.
Eine zusätzliche Routine vergleicht den aktuellen Interruptvectortable mit dem in »nodi« gespeicherten und ermittelt daraus die »verbogenen« Vektoren. Eine Druckroutine schließlich rundet das Ganze ab und bringt die derzeitige RAM-Konfiguration zu Papier.
Beim Arbeiten mit Quick-DOS ergeben sich Schwierigkeiten. Entfernt man Quick-DOS mit »clean«, kommt der Rechner nicht wie erwartet zum MS-DOS-Prompt, sondern bleibt bei dem mittlerweile entfernten Quick-DOS. Dieses Problem hängt mit dem vorher schon angeschnittenen »terminate vector« zusammen. Trotz der Restaurie-

rung der Vektoradressen im Interruptvectortable kommt das System hier gründlich durcheinander. Was ist geschehen? MS-DOS kopiert am Ende eines Programmes die Adresse des »terminate vector« im PSP in den Interruptvectortable. Die PSP-Rücksprungadresse von »clean« zeigt demnach auf Quick-DOS und nicht auf den Kommandoprozessor von MS-DOS. Wenn »clean« nach dem Ende zurückspringt, steht der Rechner buchstäblich im »Nichts«.

## Man unterscheidet drei Arten der Speicherverwaltung

Durch explizites Setzen dieser Adresse läßt sich dieser Fehler beinahe korrigieren. Beinahe – weil im PSP noch eine andere Art von Rücksprungadresse vorhanden ist, die als »parents« bezeichnet wird und allerdings meist ungenutzt bleibt.

Auch diese Adresse zeigt auf Quick-DOS, so daß sie, ähnlich wie beim »terminate vector« beschrieben, explizit neu gesetzt werden muß. Wollen Sie innerhalb von Quick-DOS einige Programme entfernen, dürfen Sie das System nicht neu initialisieren, da sonst die Parents-Adresse nicht mehr stimmt. Hierzu ein konkretes Beispiel: Sie laden »nodi« in der Datei »autoexec.bat«, und darauf folgend installieren Sie ein residentes Programm im Stil von »scrprot«. Anschließend rufen Sie Quick-DOS auf und sind damit in einer neuen »Kommandoebene« gelandet. Von hier aus installieren Sie wieder »nodi« und dann Sidekick. Rufen Sie nun »clean« auf, so werden Ihnen »nodi« und Sidekick zur Freigabe angeboten. Nach der Freigabe fragt »clean« an, ob es das System neu setzen soll. Diese Frage müssen Sie hier mit »Nein« beantworten, da Sie innerhalb der »Kommandoebene« geblieben sind. Beim erneuten Aufruf von »clean« bietet es Ihnen Quick-DOS, »scrprot« sowie das erste »nodi« an.

Da Sie bei der Freigabe von Quick-DOS die Kommandoebene verlassen, müssen Sie das System unbedingt neu konfigurieren. Um Verwirrungen zu vermeiden, schlägt das Programm die jeweils richtige Antwort vor. Trotzdem bleibt Ihnen noch die Wahl, das Rücksetzen des Programms zu veranlassen oder nicht. Allgemein kann festgestellt werden, daß ein Neusetzen des Systems dann notwendig ist, wenn die Adresse des »PSP-Terminatevector« sehr stark (um mehr als einen Paragraphen) von der des Interrupt 22hex abweicht.

Wenn Sie häufig mit speicherresidenten Applikationen arbeiten, leisten Ihnen diese Programme gute Dienste. Ist der Speicher zu knapp für eine Anwendung, so entfernen Sie einfach ein Utility aus dem Speicher und setzen Ihre Arbeit fort.

(Christian Büchel/gh)

```
[B8A]    0: code segment para public  'code'
[000]    1:
[DL1]    2: org 100h
[HF4]    3: assume cs:code
[FW4]    4: start: jmp anfang
[000]    5:
[915]    6: liste db 400h dup (?)
[000]    7:
[HF4]    8: assume ds:code
[X67]    9: anfang:xor cx,cx          ;cx=0
[645]   10:       mov es,cx           ;es=0
[Z96]   11: hier:  mov di,cx          ;di=0
[GB8]   12:       mov ax,es:[di]      ;0000:0000 nach AX
[255]   13:       mov bx,cx           ;bx=0
[TPB]   14:       add bx,offset liste ; bx=offset list
          e+0
[2WC]   15:       mov [bx],ax         ;ax([0000:0000]) na
          ch offset liste+0
[3G4]   16:       inc cx              ;cx=1
[XMB]   17:       cmp cx,400h         ;schon 400h Bytes k
          opiert ?
[GFL]   18:       jnz hier          ;wenn nicht zu hier
          bis [0:400h] nach offset liste+400h
[P52]   19:       mov ah,9
[KRF]   20:       mov dx,offset text  ;Installationsst
          ring ausgeben
[T02]   21:       int 21h
[WME]   22:       mov dx,offset anfang+1  ;bis 'anfang
          ' resident machen
[X12]   23:       int 27h
[YWB]   24:       text  db 13,10,' NODI V1.0  installi
          ert',13,10,'$'
[ML2]   25: code ends
[ZR2]   26: end start
```

**Listing 1. Das Assemblerprogramm »nodi.asm«**

```
[000]    0:
[000]    1:
[ET4]    2: program clean;
[000]    3:
[000]    4:
[RH7]    5: (* (c) Christian Büchel    *)
[000]    6:
[000]    7:
[P21]    8: uses
[7Z2]    9:   Printer,
[LT0]   10:   Crt,
[7V0]   11:   Dos;
[000]   12:
[P28]   13: const counter      : byte       = 0;
[X66]   14:       name         : string[25] = '';
[AE6]   15:       nodicount    : word       = 0;
[EDE]   16:       esc          = #$1b;    (* Escape Steu
          erseqenz für Drucker*)
[0DG]   17:       formfeed     = #12;     (* form feed S
          teuerseqenz für Drucker*)
[WHG]   18:       unten        = '0 : Suche von hoher na
          ch niedriger Adresse';
[PMF]   19:       oben         = '2 : Suche von niedrige
          r nach hoher Adresse';
[CCE]   20:       best         = '1 : Suche nach dem gün
          stigsten Platz      ';
[A21]   21: type
[ST6]   22:        string4     = string[4];
[000]   23:
[974]   24:        list        = record
[ZX5]   25:                      kennung             : ch
          ar;
[9V7]   26:                      cler,segment,owner : wo
          rd;
[9H6]   27:                      laenge              : lo
          ngint;
[1L7]   28:                      identy              : st
          ring[25];
[000]   29:
[BZ7]   30:                case  prog                : bo
          olean of
[789]   31:                      true                : (e
          nv,langenv : word)
[TU0]   32:                      end;
[000]   33:
[EM0]   34: var
[000]   35:
[0T6]   36: dummy                                    : char;
[07E]   37: ausg                                     : array[
          1..100] of list; (*max. 100 MCB's*)
[58G]   38: vec_table                                : array[
          0..$400] of byte;(*evtl. erhöhen*)
```

```
[Y4E]   39: norm                                    : array[
          0..$200] of word absolute 0:0;
[FFK]   40: intcheck                                : array[
          0..$200] of word absolute vec_table;
[S67]   41: str4                                    : string
          4;
[GH8]   42: reg                                     : regist
          ers;
[K7A]   43: mcbseg,lang,new_mcbadr,segnodi,go,
[1BB]   44: dosinfofs,dosinfseg,intvec,abhier,
[7LA]   45: ende,maxcounter,zeile                   : word;
[C28]   46: adder,int_nr                            : byte;
[MAA]   47: found,intchange                         : boolea
          n;
[000]   48:
[FF3]   49: (**********************)
[000]   50:
[7TL]   51: function hex(zahl : word) : string4;(* Wand
          elt Dezimalzahlen in
[FH7]   52:                                           Hexa
          dezimalstring um!   *)
[9R4]   53: var a,z       : word;
[U14]   54:     s         : string4;
[000]   55:
[BH1]   56: begin
[Z21]   57: s:='';
[NW9]   58: for a:=1 to 4 do    (* 4-stellige Zahl *)
[BH1]   59:  begin
[U44]   60:   z:= zahl and 15;
[BB5]   61:   if z >9 then z :=z + 7;
[VZ3]   62:   s:=chr(z+48)+s;
[249]   63:   zahl:=zahl shr 4; (* =zahl div 16 *)
[TU0]   64:  end;
[000]   65:
[282]   66: hex:=s;
[TU0]   67: end;
[000]   68:
[747]   69: (****************************************
          *******)
[ERC]   70: procedure datenholen  (var mcbseg: word);
[P22]   71: label 11;
[000]   72:
[BH1]   73: begin
[5YD]   74:  inc(counter);                            (* cou
          nter:=succ(counter)*)
[GX6]   75:  with ausg[counter] do
[BH1]   76:   begin
[TV1]   77:    env:=0;
[993]   78:    langenv:=0;
[483]   79:    prog:=false;
[XMK]   80:    cler :=mcbseg+1;                       (*Adres
          se des freizugebenden Speichers *)
[U5E]   81:    segment:=mcbseg;                       (*MCB s
          egment                              *)
[DBH]   82:    laenge:=memw[mcbseg:3]*16;             (*Länge
           des Besitzers                      *)
[94M]   83:    owner:=memw[mcbseg:1];                 (*PSP-A
          dresse des zugehörigen Programms*)
[0HF]   84:    kennung:=chr(mem[mcbseg:0]);           (* 'M'
          oder 'Z'                            *)
[H2M]   85:    if memw[mcbseg:$10]=$20CD then         (*Anfan
          gskennung PSP: CD 20 (int 32hex)*)
[XXE]   86:     begin                                 (*das h
          eißt es handelt sich um ein     *)
[S1E]   87:      prog:=true;                          (* Prog
          ramm => (prog:=true)            *)
[WYL]   88:      env:=memw[mcbseg:$3C];               (*Envir
          onmentaddresse des Programms    *)
[BBK]   89:      langenv:= memw[env-1:3]*16;          (*Länge
           des Environments               *)
[TU0]   90:     end;
[TU0]   91:   end;
[3N3]   92:  found:=false;
[2R2]   93:  abhier:=0;
[1Z1]   94:  ende:=0;
[YSB]   95:  for go:=0 to ausg[counter].langenv do
[BH1]   96:   begin
[MHM]   97:    if (mem[ausg[counter].env:go] =$00)   an
          d (found) and (go>abhier+1) then
[BH1]   98:     begin
[PXD]   99:      ende:=go;                                (
          *Pfadende wird von DOS mit 00
[953]  100:
                markiert *)
[UV1]  101:      goto 11;
[TU0]  102:     end;
[N1L]  103:     if memw[ausg[counter].env:go]=$0001 the
          n  (*Pfadanfang mit 01 00       *)
[BH1]  104:      begin
[K13]  105:       abhier:=go;
[PNG]  106:       found:=true;   (* Flag, das der Suchr
          outine für das Ende
[H0D]  107:                                des Namens mitte
          ilt ob überhaupt schon die
[WUA]  108:                                Anfangskennung g
          efunden wurde.     *)
[TU0]  109:      end;
[TU0]  110:   end;
[1A0]  111:  11:
```

```
[TAN]  112:  for go:=abhier to ende do      (*von Pfada
          nfang bis Ende Zeichen einlesen*)
[BH1]  113:   begin
[LRM]  114:    if (mem[ausg[counter].env:go]>31) and (m
          em[ausg[counter].env:go]<95)
[0TE]  115:    then name:=name+chr(mem[ausg[counter].en
          v:go])
[H5M]  116:    else name:=name+'';       (*nur ASCII-Ze
          ichen von 32-94(Großbuchstaben)*)
[TU0]  117:   end;
[EB8]  118:  ausg[counter].identy:=name;
[C3N]  119:  if (ausg[counter].identy='') and (ausg[cou
          nter].prog=true) then
[NUF]  120:  ausg[counter].identy:='Kein Programmame ge
          funden';
[7Z1]  121:  name:='';
[DMD]  122:  if (pos('NODI.COM',ausg[counter].identy) <
          >0)
[VAE]  123:  then                    (* ist NODI.COM in
          Pfadname enthalten ? *)
[35L]  124:   begin                   (* die Frage nach
          memw[..owner:$100] ist sehr wichtig,
[9YB]  125:                                da z.B. KEYBGR.E
          XE seinen Namen 'löscht'
[NKE]  126:                                und einfach der
          Name vom nächsten residenten
[6TC]  127:                                Programm von DOS
           dafür eingesetzt wird.
[4XD]  128:                                RELEASE/CLEAN wü
          rden ihre Daten dann in KEYBGR    *)
[EPN]  129:    if (memw[ausg[counter].owner:$100]=$00E9
          ) (* suchen,ein Absturz wäre die *)
[TP9]  130:                          (* Folge.E9 00 04='
          jmp anfang'in NODI*)
[FUB]  131:    and (mem[ausg[counter].owner:$102]=$04)
          then
[BH1]  132:     begin
[1XG]  133:      nodicount:=counter;                  (* W
          enn ja , welche lfd. Nummer ? *)
[7UK]  134:      segnodi:=ausg[counter].owner;        (* W
          elche  Segmentadresse ?       *)
[DL0]  135:     end
[U9L]  136:     else ausg[counter].identy:='Kuckucksei'
          ; (* siehe oben (KEYBGR)*)
[TU0]  137:   end;
[AMH]  138: maxcounter:=counter-1;                  (* Anzahl
           aller MCB-Einträge       *)
[TU0]  139: end;
[YK9]  140: (****************************************
          **********************)
[000]  141:
[KP5]  142: procedure druck;
[X22]  143: label 22;
[000]  144:
[BH1]  145: begin
[7GN]  146: assign(lst,'PRN');     (* Gegebenenfalls an
          deren Namen für 'PRN' einsetzen *)
[A34]  147: rewrite(lst);
[58K]  148: writeln(lst,'  MCB    Owner    Segment    K
          ennung     Länge (Bytes)    Name ');
[ZEE]  149: writeln(lst,'------------------------------
          ---------------------------------');
[508]  150: for go:=1 to maxcounter do
[BH1]  151:  begin
[TX4]  152:   with ausg[go] do
[BH1]  153:    begin
[DG4]  154:     write (lst,'  ',go:2);
[F87]  155:     write (lst,'     ',hex(owner));
[BX7]  156:     write (lst,'     ',hex(segment));
[797]  157:     write (lst,'          ',kennung );
[TC7]  158:     write (lst,'      ',laenge:6,'    ');
[W96]  159:     writeln(lst,identy);
[5T3]  160:     if prog then
[BH1]  161:      begin
[YUC]  162:       write(lst,esc,'E');                 (* andere
           Schrift wählen*)
[7S9]  163:       writeln (lst,'                    ' ,h
          ex(env),
[BPA]  164:                          '                ',lang
          env:6,'      Environment!');
[S8D]  165:       write(lst,esc,'F');                 (* wieder
           normale Schrift  *)
[TU0]  166:      end;
[TU0]  167:    end;
[TU0]  168:  end;
[SPD]  169: writeln (lst,'         Noch ',memavail,' By
          tes frei !');
[1K4]  170: writeln (lst);
[BCR]  171: writeln (lst,'Das nächste Programm wird nac
          h  ',hex(prefixseg),':0000 geladen');
[1K4]  172: writeln (lst);
[1K4]  173: writeln (lst);
[1K4]  174: writeln (lst);
[E5M]  175: if intchange then   (* Flag wird von Intche
          ck gesetzt wenn Interrupt
[5X5]  176:                               verändert wurde
          *)
[BH1]  177:  begin
[7PP]  178:   writeln(lst,'Folgende Interruptvektoren w
```

```
          urden seit Nodi verändert  :');
[74D]  179:    writeln(lst,'-------------------------
          ----------------------------');
[XRF]  180:    writeln(lst,'   Nr.            Alte Adress
          e          Neue Adresse       ');
[234]  181:    writeln(lst);
[MLF]  182:    move (memw[segnodi:$103],intcheck,sizeof(
          intcheck));
[7UF]  183:    for int_nr:=0 to $FF do     (*Insgesamt 2
          56 Interrupts *)
[ARF]  184:     begin                      (*Ab Offset 103h
           sind die alten Interrupt-
[L29]  185:                                          adressen
          in NODI gespeichert.
[ZC8]  186:                                          jede Adre
          sse in 4 Bytes (z.B.
[4V4]  187:                                          34 12 78
          56 = 5678:1234        *)
[000]  188:
[X17]  189:      intvec:=int_nr+int_nr;
[EUA]  190:      if int_nr in [0,$1B,$23,$24,$75] then g
          oto 22;
[KLB]  191:      if (norm[intvec]<>intcheck[intvec])
[UGC]  192:      and (norm[intvec+1]<>intcheck[intvec+1]
          )
[54L]  193:      then          (*norm sind die derzeit a
          ktuellen Interrupt Adressen*)
[49F]  194:       begin         (*intcheck die in NODI g
          espeicherten          *)
[TXC]  195:        write(lst,'   ',copy(hex(intvec div 2
          ),3,2),'              ',
[ZE7]  196:                    hex(intcheck[intvec+1]));
[R1L]  197:        write(lst,':',hex(intcheck[intvec]),'
                     ',hex(norm[intvec+1]));
[3LA]  198:        writeln(lst,':',hex(norm[intvec]));
[G1C]  199:                  (* Das Programm (Turbo-Pasca
          l) selbst legt
[0XB]  200:                  Interruptvektoren  Nr. 00
          ,1B,23,24,75hex um,
[VHE]  201:                  die am Ende aber wieder z
          urückgebogen werden !*)
[000]  202:
[TU0]  203:       end;
[4A0]  204:      22:
[TU0]  205:     end;
[DL0]  206:   end
[W5L]  207:   else  writeln(lst,'Kein Interruptvektor se
          it NODI verbogen');
[316]  208:   write(lst,formfeed);
[8X2]  209: close(lst);
[TU0]  210: end;
[Z88]  211: (*****************************************
          ***************)
[9Y5]  212: procedure output;
[000]  213:
[BH1]  214: begin
[NM2]  215: clrscr;
[9Z2]  216: writeln;
[YBE]  217: write ('   Pointer auf DOS-INFO :      ',hex
          (dosinfseg));
[YN8]  218: writeln (':',hex(dosinfofs));
[9Z2]  219: writeln;
[NN3]  220: gotoxy(30,12);
[BU7]  221: writeln('Taste drücken');
[S05]  222: dummy:=readkey;
[R04]  223: window(1,1,80,25);
[NM2]  224: clrscr;
[43K]  225: writeln('  MCB    Owner   Segment   Kennung
             Länge (Bytes)  Name ');
[FZD]  226: writeln('-----------------------------------
          ---------------------------------');
[9Z2]  227: writeln;
[UC3]  228: gotoxy(1,24);
[8LN]  229: writeln('Bitte Taste drücken      Mit Contr
          ol-C raus   * Env * : Environment ');
[Z04]  230: window(1,5,80,23);
[NM2]  231: clrscr;
[AB2]  232: zeile:=0;
[508]  233: for go:=1 to maxcounter do
[BH1]  234:  begin
[TX4]  235:   with ausg[go] do
[BH1]  236:    begin
[E44]  237:     write ('  ',go:2);
[M46]  238:     write ('     ',hex(owner));
[8P6]  239:     write ('     ',hex(segment));
[686]  240:     write ('       ',kennung );
[6H6]  241:     write ('     ' ,laenge:6,'     ');
[UH5]  242:     writeln(identy);
[N03]  243:     inc(zeile);
[5T3]  244:     if prog then
[BH1]  245:      begin
[6Z9]  246:       writeln ('                    ' ,hex(en
          v),'              ',
[DP5]  247:                 langenv:6,'     * ENV * ');
[9Z2]  248:       writeln;
[BE3]  249:       inc(zeile,2);
[TU0]  250:      end;
[KT4]  251:     if zeile >=16 then
[BH1]  252:      begin
```

Bestellungen bitte nur schriftlich an

**ABACOMP GmbH**

Abt. DE, Kransberger Weg 24, 6000 Frankfurt/M. 50
Ladenverkauf: Mo-Fr 10-12 + 14-18h, Heerstr.149, 6 Ffm90
Technische Auskunft: Mo-Sa 8-9.30h unter (069) 76 30 39

**Die neuen PC´s sind da: ABACO® PC/2**

XT/AT-kompatible Rechner der zweiten Preis/Leistungs-Generation:

**ABACO 16HS**, XT-kompatibel, 256KB RAM, 1 Diskettenlaufwerk 360KB, 4,77/9,54MHz, V-20-Prozessor (Norton SI: 3,7), Color-Grafik-Karte, Druckerschnittstelle, Profi-Tastatur, Stromverbrauch Niederspannung ca. 15W, Netzteil 135W (daher 120W Reserveleistung), 200 Tage ABACOMP-Garantie ..... **997,50**

**ABACO 16-286**, AT-kompatibel, 512KB RAM, 1 Diskettenlaufwerk 1,2MB, 6/12MHz, Color-Grafik-Karte, Druckerschnittstelle, Profi-Tastatur, Stromverbrauch Niederspannung ca. 40W, Netzteil >150W (daher >110W Reserveleistung), 200 Tage ABACOMP-Garantie ..... **1881,-**

**ABACO AT16**, AT-kompatibel, 2MB RAM, 1 Diskettenlaufwerk 1,2MB, 6/12MHz ohne wait-states (Norton SI: 13,7), Color-Grafik-Karte, Komfort-Tastatur, Drucker- und DFÜ-Schnittstelle, Gameport, Festplatte 60MB mit superschnellem Controller, 2 Jahre ABACOMP-Garantie ..... **3990,-**

Alle Geräte sind in Deutschland gefertigt und entsprechen den postalischen Bestimmungen.

***NEU!* Endlich eine Alternative zur Festplatte !**

Aufpreis für ein 5 1/4"-Diskettenlaufwerk mit 5,5MByte Speicherkapazität, mittlere Zugriffszeit 200msec., Datenübertragungsrate ca. 30KByte/sec. (d.h. in jeder Hinsicht schneller als manche Festplattenlaufwerke), liest Disketten 360KB und 1,2MB, die extrem sichere Alternative zum Festplattenlaufwerk ..... **969,-**
Aufpreis für ein 5 1/4"-Diskettenlaufwerk mit 9,9MByte, mittlere Zugriffszeit 65msec., Übertragungsrate ca. 50KByte/sec., ..... **1710,-**

**Super-Hits des Monats**

| | |
|---|---|
| Festplattenkit 30MB für XT und AT (Adaptec-Controller) | 627,- |
| Arithmetikprozessor 80387-20MHz | 1482,- |
| VGA-Karte ATI-VIP (256 aus 262000 Farben, 640x480 und 800x560 Pkt.) | 541,50 |
| ABACO VGA-Karte, alle VGA-Modi, Hardware-Zoom 0,1sec. 1024x768 Pkt. | 627,- |
| PHILIPS 12"-TTL-Monitor, bernstein | 114,- |
| Monitor Multisync, schwarz/weiß | 570,- |
| TVM-Multisync, Bildfrequenz bis 75Hz, bis 800x600 Pkt. | 997,50 |
| ABACO-Multisync, Bildfrequenz bis 90Hz, bis 1024x768 Pkt. | 1368,- |
| ATARI 1040STF, mit Monitor SM-124 und Maus | 1425,- |
| Drucker Star LC-10 Color, 144 Z/sec., NLQ | 741,- |
| Drucker Seikosha SL-80Ai, 24 Nadeln, 135 Z/sec., 16KB Pufferspeicher | 741,- |
| **PENTAX-Flachbettscanner DIN A4, 300dpi, 64 Graustufen, Bildbearbeitung im 300dpi-Modus, OCR-Texterkennung, mit Interface für XT/AT** | **2850,-** |

*Viele weitere Produkte auf Anfrage, Händleranfragen erwünscht.*
*Wir suchen ständig neue Mitarbeiter für Verwaltung und Verkauf.*

**C-TOOLS**

**C-terp** der prof. C-Interpreter **neu: für TURBO C nur**.....**DM 350,–**

**DeSmet C Development Package** inkl. Debugger jetzt V3.1!.....**DM 295,–**

**GraphiC** wiss. Präsentationsgrafik jetzt neue **Version 4.1**.....**DM 855,–**

**C_Graph Profi-System V2.1** Geräteunabhängige Grafik nach GKS...**DM 798,–**

**Scientific Graphics Library** v. Cobra inkl. Pop-Ups f. EGA + Hercules..**DM 495,–**

**CDOC** Dokumentationssystem für C-Programme.....**DM 285,–**

**Vance C-lib** Window Bibliothek (UNIX "curses" kompatibel).....**DM 295,–**

**MID** Treiber für beliebig viele V24-Schnittstellen.....**DM 285,–**

**PC-lint** C Syntax- und Semantikprüfung über mehrere Files.....**DM 399,–**

**MKS-Toolkit** UNIX-Utilities inkl. AWK, Korn-Shell, VI, YACC etc.....**DM 495,–**

**BTREE + ISAM** neue Version 2.6 **DM 350.-**
**jetzt mit:** File Recovery Utility, alle Datentypen als Index
**ab sofort: als qbFiler speziell für MS-QuickBasic 4.0 DM 245.-- !!!**

**Viele Produkte mehr: Info anfordern! Keine zusätzlichen Versandkosten!**

**KESSLER Softwareentwicklung** Mitteldorfstr.17 3400 Göttingen Tel. 0551-792488

```
[S05]  253:        dummy:=readkey;
[NM2]  254:        clrscr;
[AB2]  255:        zeile:=0;
[TU0]  256:      end;
[TU0]  257:    end;
[TU0]  258:  end;
[S05]  259: dummy:=readkey;
[9Z2]  260: writeln;
[8NC]  261: writeln ('Noch ',memavail,' Bytes frei !');
[9Z2]  262: writeln;
[KSN]  263: write ('Das nächste Programm wird nach  ',h
       ex(prefixseg),':0000 geladen');
[VL4]  264: dummy:=readkey
[PNF]  265: end;                                    (* Pr
       efixseg ist in Unit Dos von Turbo
[Y18]  266:                                          4.
       0 definiert (=PSP-Segment)  *)
[000]  267:
[U99]  268: (*****************************************
       **********************)
[000]  269:
[AWC]  270: procedure newmcb (var  new_mcbadr : word);
[000]  271:
[CLH]  272: begin                                 (* Die MCB
       's sind vorwärts verkettet die   *)
[GLP]  273: lang:=memw[new_mcbadr:3];              (* Adresse
       des nächsten MCB's errechnet sich*)
[PFR]  274: new_mcbadr:= new_mcbadr+lang+1;  (* aus alt
       em MCBsegment+Länge+1. Länge des *)
[47H]  275: end;                                    (* MCB   s
       teht an Offset 3 des MCBsegments *)
[U99]  276: (*****************************************
       **********************)
[000]  277:
[5P4]  278: procedure get;
[000]  279:
[BH1]  280: begin
[NM2]  281: clrscr;
[6B3]  282: counter:=0;
[RAG]  283: reg.ax:=$5200;      (*Undokumentierte DOS-Fu
       nktion 52h                *)
[Y9H]  284: msdos(reg);         (*übergibt in ES:BX Poin
       ter auf DOS-Info-Block. *)
[E2M]  285: dosinfseg:=reg.es; (*Das Segment des ersten
        MCB steht am Offset-2  *)
[BE5]  286: dosinfofs:=reg.bx;
[0F9]  287: mcbseg:=memw[reg.es : reg.bx-2];
[HX5]  288: datenholen (mcbseg);
[9T5]  289: new_mcbadr:=mcbseg;
[X52]  290:  repeat
[NU5]  291:  newmcb (new_mcbadr);
[RU7]  292:  datenholen (new_mcbadr);
[K5D]  293:  until  (mem[new_mcbadr :0])=90;     (* Kenn
       ung = 'Z' *)
[TU0]  294: end;
[618]  295: (*****************************************
       ************)
[000]  296:
[ZT5]  297: procedure rescue;
[BH1]  298: begin
[TV0]  299: get;
[R04]  300: window(1,1,80,25);
[NM2]  301: clrscr;
[F65]  302: if nodicount=0 then
[BH1]  303:  begin
[9Z2]  304:   writeln;
[Y5B]  305:   writeln('Nodi nicht installiert');
[7C1]  306:   exit;
[TU0]  307:  end;
[UHN]  308: if nodicount <= maxcounter  then   (*Ist üb
       erhaupt was residentes da ?*)
[BH1]  309:  begin
[AB2]  310:   zeile:=0;
[YNK]  311:   writeln('Folgende Programme können freige
       geben werden :');
[9Z2]  312:   writeln;
[RMC]  313:   for go := maxcounter downto nodicount do
[BH1]  314:    begin
[8YF]  315:     writeln(hex(ausg[go].cler),' ( ',ausg[g
       o].identy,' ) ');
[N03]  316:     inc(zeile);
[PZ7]  317:     if ausg[go].env<>0 then
[BH1]  318:      begin
[LDB]  319:       writeln (hex(reg.ax),' (Environment)
       ');
[N03]  320:       inc(zeile);
[TU0]  321:      end;
[Z05]  322:     if zeile>18 then
[BH1]  323:      begin
[AB2]  324:       zeile:=0;
[9Z2]  325:       writeln;
[37K]  326:       writeln('Um weiterzumachen beliebige
       Taste drücken ..... ');
[S05]  327:       dummy:=readkey;
[TU0]  328:      end;
[TU0]  329:    end;
[9Z2]  330:  writeln;
[0CD]  331:  write('Soll Nodi entfernt werden ?
       (Y/N)  ');
```

```
[X52]  332:   repeat
[S05]  333:   dummy:=readkey;
[VTC]  334:   until (upcase(dummy)='N') or (upcase(dumm
       y)='Y');
[UB7]  335:  writeln(upcase(dummy));
[07F]  336:  if upcase(dummy)='N' then adder:=1 else ad
       der:=0;
[CLK]  337:  if (upcase(dummy)='N') and (maxcounter-nod
       icount=0) then halt;
[N3E]  338:  write('Soll ich  Programme freigeben ?
       (Y/N)  ');
[X52]  339:   repeat
[S05]  340:   dummy:=readkey;
[VTC]  341:   until (upcase(dummy)='N') or (upcase(dumm
       y)='Y');
[UB7]  342:  writeln(upcase(dummy));
[55A]  343:  if upcase(dummy)='N' then exit;
[F45]  344:  for go:= 0 to $400 do
[NEM]  345:  vec_table[go]:=mem[segnodi:$103+go]; (* Ab
       Offset 0103hex steht in NODI.COM
[N6B]  346:
       der 'konservierte' Interruptvector-
[6F5]  347:
       table (400hex Bytes) !*)
[000]  348:
[ZWF]  349:  inline ($FA);                          (*CLI
       (Keine Interrupts erlauben!*)
[29P]  350:  move (vec_table,mem[0:0],sizeof(vec_table)
       );          (*Vektoren zurücksetzen*)
[XRG]  351:  inline ($FB);                          (*STI
       Ok Interrupts wieder erlaubt*)
[000]  352:
[PKE]  353:  for go := maxcounter downto nodicount+adde
       r do
[BH1]  354:   begin
[7H9]  355:    reg.ax:=$4900;                (* MEMORY F
       REIGEBEN *)
[NTP]  356:    reg.es:=ausg[go].cler;        (*Segmentad
       resse des zu entfernenden Programms*)
[KZ2]  357:    msdos(reg);
[AR4]  358:    if reg.ax=$0009 then
[BH1]  359:     begin
[YLG]  360:      writeln (^G,'ACHTUNG Speicherfreigabe
       nicht geklappt');
[TU0]  361:     end;
[PEP]  362:    if ausg[go].env<>0 then       (*Wenn AX=0
       009hex  dann ist Fehler aufgetreten*)
[BH1]  363:     begin
[XD3]  364:      reg.ax:=$4900;
[5H6]  365:      reg.es:=ausg[go].env;
[KZ2]  366:      msdos(reg);
[TU0]  367:     end;
[TU0]  368:   end;
[TU0]  369:  end;
[9Z2]  370: writeln;
[F1R]  371: writeln('PSP-Terminate   Vektor     zeigt a
       uf  ',hex(memw[prefixseg:$C]),' : ',
[LE7]  372:          hex(memw[prefixseg:$A]));
[LYL]  373: writeln('Terminate Vektor (Int 22h) zeigt a
       uf  ',hex(memw[0:$8A]),' : ',
[ZH4]  374:          hex(memw[0:$88]));
[9TP]  375: writeln('Segment des ausführenden Programms
       :  ',hex(memw[prefixseg:$16]));
[9Z2]  376: writeln;
[60D]  377: if abs(memw[0:$8A]-memw[prefixseg:$C])>2 th
       en
[ZCL]  378: writeln('Ich würde vorschlagen das System n
       eu zu setzten !')
[S2L]  379: else writeln('Das System sollte nicht neu g
       esetzt werden !');
[ALE]  380: write('Soll  System  neu  gesetzt  werden
       ? (Y/N)  ');
[X52]  381:  repeat
[S05]  382:  dummy:=readkey;
[VTC]  383:  until (upcase(dummy)='N') or (upcase(dummy
       )='Y');
[UB7]  384: writeln(upcase(dummy));
[9Z2]  385: writeln;
[NS7]  386: if upcase(dummy)='Y' then
[BH1]  387:  begin
[KSN]  388:   memw[prefixseg:$A]:=memw[0:$88];      (* W
       ieder herstellen der Rücksprung- *)
[R1K]  389:   memw[prefixseg:$C]:=memw[0:$8a];      (* r
       outinen  PSP INT 22              *)
[000]  390:
[D0M]  391:   memw[prefixseg:$16]:=memw[0:$8A];     (* S
       egment des Laders ('parents')    *)
[TU0]  392:  end;
[LYL]  393: writeln('Terminate Vektor (Int 22h) zeigt a
       uf  ',hex(memw[0:$8A]),' : ',
[ZH4]  394:          hex(memw[0:$88]));
[F1R]  395: writeln('PSP-Terminate   Vektor     zeigt a
       uf  ',hex(memw[prefixseg:$C]),' : ',
[LE7]  396:          hex(memw[prefixseg:$A]));
[9TP]  397: writeln('Segment des ausführenden Programms
       :  ',hex(memw[prefixseg:$16]));
[9Z2]  398: writeln;
[S05]  399: dummy:=readkey;
[4CN]  400: (* An Offset 0Ahex und 0Chex des PSP steht
```

```
          die Rücksprungadresse zum Auf-
[AKR]  401:    rufer.Da dieses 'Mutterprogramm' aber en
          tfernt wurde muß diese PSP-Adresse
[XWM]  402:    neu gesetzt werden.Die richtige Adresse
          steht im restaurierten
[46L]  403:    Interruptvektortable (INT 22hex) an Spei
          cherstelle 0000:0088 hex
[K7G]  404:    (4 Bytes ,je 2 für Segment und Offset) s
          iehe auch unten !
[91N]  405:    An Offset 16hex steht die Segmentadresse
           des Programmes von dem
[PSM]  406:    aus das laufende Programm gestartet wurd
          e('parents').Das ist bei
[7PN]  407:    vielen speicherresidenten Programmen der
           Fall,die selbst Programme
[CVK]  408:    ausführen können z.B. Quick-Dos.Diese Ad
          resse stimmt nach der
[AWR]  409:    Speicherfreigabe aber nicht mehr, wenn d
          as 'Mutterprogramm'auch entfernt
[59M]  410:    wurde. Dann muß diese Stelle des PSP  di
          e Segmentadresse des vor
[M5M]  411:    dem Laden von NODI.COM aktuellen Termina
          tevektor(INT 22hex) bekommen.
[KEM]  412:    Diese steht im (restaurierten)Interruptv
          ektortable an Offset 8Ahex
[1V1]  413:    (0000:008A).
[43D]  414:    WENN das Mutterprogramm NICHT entfernt w
          ird
[7RC]  415:    soll die Adresse NICHT neu gesetzt werde
          n.
[513]  416:    Allgemein:
[9XF]  417:    Ist das Segment des Laders nur wenig (ca
          .1 Paragraph)
[E78]  418:    verschieden von dem des
[1GF]  419:    INT 22h Vektors ist, dann das System NIC
          HT neu setzen.
[3UK]  420:    Besteht ein großer Unterschied zwischen
          beiden Adressen,
[WMB]  421:    MUSS das System neu gesetzt werden !!!!
[W30]  422: *)
[000]  423:
[000]  424:
[TU0]  425: end;
[U43]  426: (********************)
[7H7]  427: procedure mem_control;
[000]  428:
[BH1]  429: begin
[253]  430: gotoxy(1,6);
[77N]  431: writeln ('Die dezeitige Art der Speicherver
          teilung von DOS ist : ');
[WD3]  432: reg.ax:=$5800;
[KZ2]  433: msdos(reg);
[8C4]  434: case reg.ax of
[XG5]  435:  0 : writeln(unten);
[G55]  436:  1 : writeln(best);
[Y35]  437:  2 : writeln(oben);
[TU0]  438: end;
[9Z2]  439: writeln;
[9RE]  440: writeln ('Speicherverteilung neu setzen ? '
          );
[375]  441: writeln(unten);
[DL4]  442: writeln(best);
[KH4]  443: writeln(oben);
[9Z2]  444: writeln;
[8HH]  445: write ('Zum Ändern Zahl eingeben, sonst and
          ere Taste !   ');
[S05]  446: dummy:=readkey;
[LAB]  447: if (ord(dummy)<51) and (ord(dummy)>47) then
[BH1]  448:  begin
[295]  449:   writeln(dummy);
[176]  450:   reg.bx:=ord(dummy)-48;
[1E3]  451:   reg.ax:=$5801;
[KZ2]  452:   msdos(reg);
[TU0]  453:  end;
[TU0]  454: end;
[U43]  455: (********************)
[PH7]  456: procedure checkvector;
[X22]  457: label 22;
[BH1]  458: begin
[TV0]  459: get;
[LS2]  460: output;
[NM2]  461: clrscr;
[Z7D]  462: if nodicount=0 then
              (* kein NODI da *)
[BH1]  463:  begin
[3CC]  464:   writeln('  NODI ist nicht installiert  ')
          ;
[7C1]  465:   exit;
[TU0]  466:  end;
[KL5]  467: intchange:=false;
[R04]  468: window(1,1,80,25);
[NM2]  469: clrscr;
[9Z2]  470: writeln;
[5LP]  471: writeln('Folgende Interruptvektoren wurden
          seit Nodi verändert  :');
[T1C]  472: writeln('-------------------------------
          ----------------------');
[L9F]  473: writeln('   Nr.          Alte Adresse
            Neue Adresse       ');
```

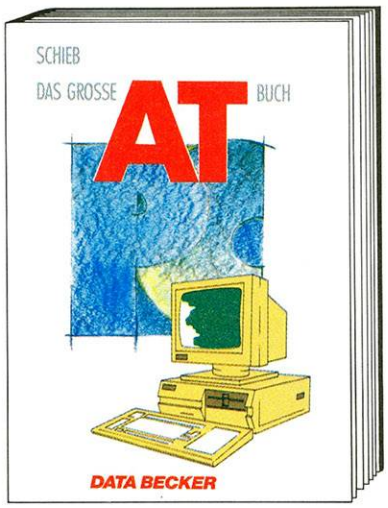
```
[9Z2]  474: writeln;
[MLF]  475: move (memw[segnodi:$103],intcheck,sizeof(in
          tcheck));
[AZ6]  476: for int_nr:=0 to $ff do
[BH1]  477:  begin
[X17]  478:   intvec:=int_nr+int_nr;
[KLB]  479:   if (norm[intvec]<>intcheck[intvec])
[M0E]  480:   and (norm[intvec+1]<>intcheck[intvec+1])
          then
[BH1]  481:    begin
[NLA]  482:     if int_nr in [0,$1B,$23,$24] then goto
          22;
[5MC]  483:     write('   ',copy(hex(intvec div 2),3,2)
          ,'          ');
[ZE7]  484:           hex(intcheck[intvec+1]));
[B4H]  485:     write(':',hex(intcheck[intvec]),'
          ',hex(norm[intvec+1]));
[91A]  486:     writeln(':',hex(norm[intvec]));
[HRL]  487:     intchange:=true;    (* aktuellen Interr
          uptvectortable mit dem *)
[BLD]  488:     22:                  (* damaligen (NODI.
          COM) vergleichen       *)
[TU0]  489:    end;
[TU0]  490:  end;
[GAG]  491: if not intchange then writeln('Kein Vektor
          verbogen  !');
[9Z2]  492: writeln;
[709]  493: write ('Liste ausdrucken (J/N) ?');
[S05]  494: dummy:=readkey;
[3Z9]  495: if upcase(dummy)= 'J' then  druck;
[NM2]  496: clrscr;
[5S3]  497: mem_control;
[TU0]  498: end;
[FF3]  499: (**********************)
[000]  500:
[YN8]  501: begin                    (***** HAUPTPROGRAMM
           ******)
[NM2]  502: clrscr;
[253]  503: gotoxy(1,6);
[9Z2]  504: writeln;
[NZM]  505: writeln ('       Programm  zum Entfernen vo
          n residenten Programmen');
[9Z2]  506: writeln;
[KC3]  507: gotoxy(1,12);
[G5N]  508: writeln ('   *  (R)escue          (Löscht r
          esidente Programme bis Nodi.com)');
[VC3]  509: gotoxy(1,15);
[WUL]  510: write   ('   *  (C)heck memory    (Zeigt ak
          tuelle Speicherbelegung)');
[S05]  511: dummy:=readkey;
[W2A]  512: if upcase(dummy)= 'R' then rescue
[EG5]  513:                       else checkvector;
[5T0]  514: end.
```

**Listing 2. Das Pascalprogramm »clean.pas«**

```
[9N2]    0: n nodi.com
[LG0]    1: a100
[2N1]    2: jmp 503
[000]    3:
[ZR0]    4: a 503
[RC2]    5: xor cx,cx
[4C2]    6: mov es,cx
[3A2]    7: mov di,cx
[TF0]    8: es:
[3V2]    9: mov ax,[di]
[LC2]   10: mov bx,cx
[TA2]   11: add bx,103
[Y63]   12: mov [bx],ax
[DX1]   13: inc cx
[GC2]   14: cmp cx,400
[YP1]   15: jnz 507
[F72]   16: mov ah,09
[5E2]   17: mov dx,527
[1A1]   18: int 21
[XD2]   19: mov dx,504
[5B1]   20: int 27
[000]   21:
[7H4]   22: e 527 0D 0A 20 4E 4F 44 49 20
[9P4]   23: e 52F 56 31 2E 30 20 20 69 6E
[WM4]   24: e 537 73 74 61 6C 6C 69 65 72
[5S2]   25: e 53F 74 0D 0A 24
[0N0]   26: rcx
[N90]   27: 443
[R30]   28: w
[H30]   29: q
```

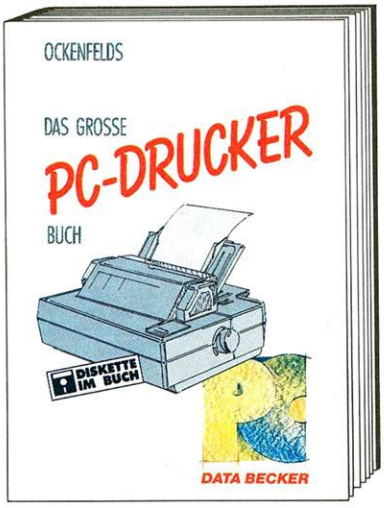
**Listing 3. »nodi.com« können Sie auch mit Debug erzeugen**

# NEUE PC

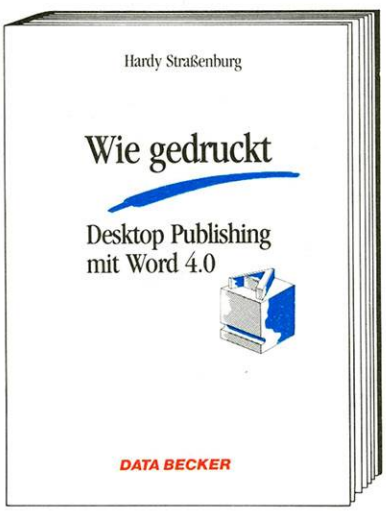



**Den PC/AT verstehen.**
In diesem Buch erfahren Sie alles, was Sie zum AT wissen wollen. Die Entwicklung, die Gemeinsamkeiten und Unterschiede zum Vorgänger PC/XT, das Installieren des Systems, die Verwendung der Betriebssysteme DOS und OS/2, die Programmierung der Tastatur, die Organisation der Festplatte, die Unterschiede der Videokarten (MDA, CGA, EGA, VGA, PGA, Hercules), alles über EMS, LIM, Windows, Speichererweiterungen und vieles andere mehr. Mit zahlreichen Abbildungen und Programmbeispielen.
**Das große AT-Buch**
**Hardcover, 390 Seiten, DM 69,–**

**Die aktualisierte Neuauflage zu einem heißen Thema.**
Kaum einer hat sich so intensiv mit dem Thema Computer-Viren beschäftigt wie Ralf Burger. In diesem Buch gibt er sein ganzes Wissen weiter. Detailliert und umfassend beschreibt er die faszinierend schreckliche Welt der Computer-Viren. Mit Programmlistings von Demonstrations-Virenprogrammen und vielen Beispielen zu möglichen Manipulationen durch Viren. Dazu zahlreiche Erfahrungsberichte von Hackern und Managern. Ein Buch, bei dem Ihnen nichts verborgen bleibt.
**Das große Computer-Viren-Buch**
**363 Seiten, DM 49,–**




**Rund um den PC-Drucker**
Nutzen Sie Ihren Drucker richtig – lesen Sie das große PC-Drucker-Buch. Hier erfahren Sie mehr über die Bedeutung der DIP-Schalter, wie ein Druckertreiber funktioniert und was Sie tun müssen, wenn der Ausdruck einmal keine Umlaute enthält. Denn das große PC-Drucker-Buch steckt voller Lösungen. Gleichzeitig dient es als hervorragendes Nachschlagewerk zu allen Druckerproblemen und -fragen. Ganz neu: mit beiliegender Diskette.
**Das große PC-Drucker-Buch**
**Hardcover, inkl. Diskette, 372 Seiten, DM 59,–**



**Der Schlüssel zur professionellen Gestaltung.**

Wie gedruckt – DTP mit Word 4.0 – hier finden Sie alles für wirklich anspruchsvolle Word-Anwendungen: Erstellen von Druckformaten für Briefe, längere Texte, Tabellen und Formulare, Serienbriefe mit Word, Arbeiten mit Makros, Übernahme von Daten aus Kalkulationsprogrammen, Fußnoten mit Word, Grafiken mit Word und und und. Natürlich mit zahlreichen Beispielen und vielen Tips aus der Praxis eines Word-Profis. Eben ein Buch für alle, die mehr über Word, aber auch über Desktop-Publishing wissen wollen. Aus der Kombination von beiden ergeben sich ganz andere Perspektiven. Dieses Buch beweist es.
**Wie gedruckt – Desktop Publishing mit Word 4.0**
**Hardcover, 265 Seiten, DM 79,–**

Wie der Ventura Publisher grundsätzlich funktioniert, ist Ihnen eigentlich klar. Aber dennoch, eine ganze Reihe von Fragen bleibt offen: Wie arbeitet man mit den Stylesheets und wie werden sie definiert? Wie lassen sich Rahmen optimal einsetzen? Welche Schriftgröße für welchen Zweck? Wie funktioniert die Arbeit mit Fußnoten, Inhaltsverzeichnis oder Index? Fragen, die Ihnen Wie gedruckt – DTP mit dem Ventura Publisher klipp und klar beantwortet. Ein echtes Anwenderhandbuch, das voller Ideen und Anregungen steckt und Ihnen auch die praktische Umsetzung vermittelt. Mit jeder Menge Tips & Tricks zur täglichen Arbeit.
**Wie gedruckt – Desktop Publishing mit Ventura Publisher**
**Hardcover, ca. 250 Seiten, DM 79,–**
**erscheint ca. 8/88**

*In Vorbereitung!*

**Das große Buch zu**

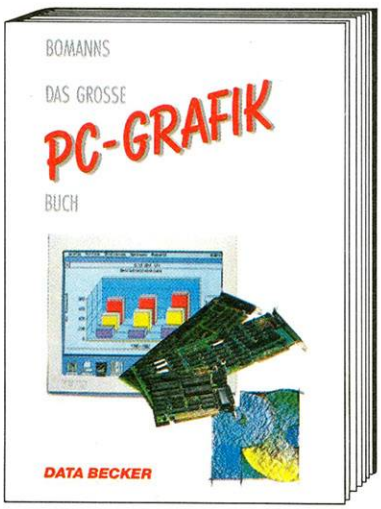
**MS-DOS 4.0**

Alles über die neueste Version von Microsofts Betriebssystem-Standard. Natürlich wieder vom MS-DOS-Profi und Bestsellerautor Jörg Schieb.

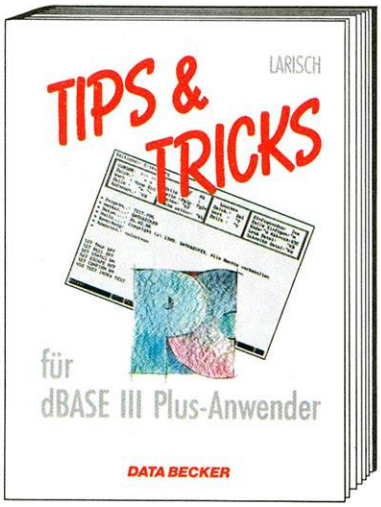
**Das große Buch zu MS-DOS 4.0**
**Bereits ab ca. Ende September lieferbar.**
**Über 500 Seiten, DM 59,–**

**DATA BECKER**
Merowingerstr. 30 · 4000 Düsseldorf · Tel. (0211) 310010

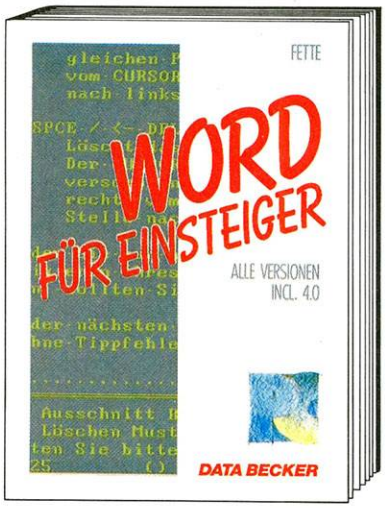
# -BÜCHER



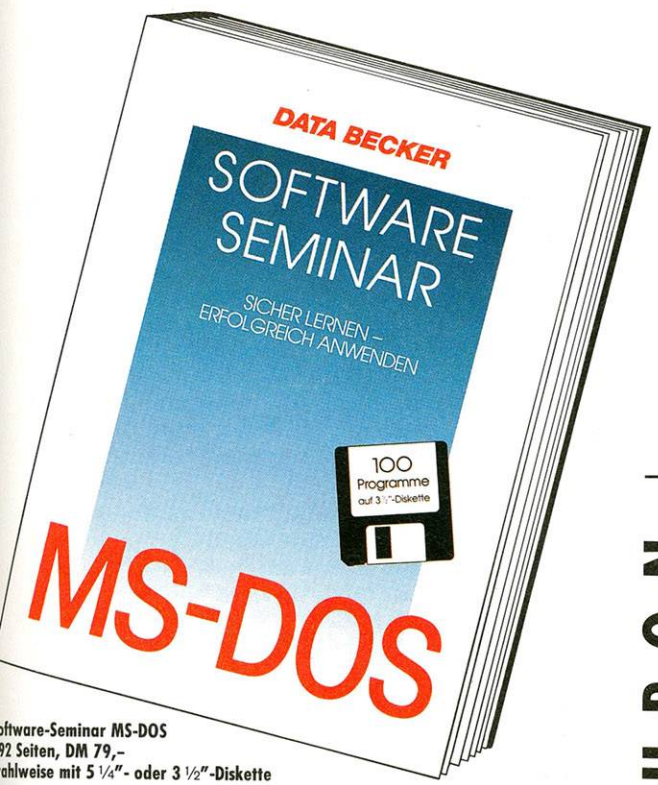

**Alles über Grafikkarten**
CGA, HGC, HRF, EGA/AGA, VGA – welche Leistungen bieten die verschiedenen Grafikkarten? Wie werden sie installiert und wie programmiert? Welche Zusammenhänge bestehen zwischen den einzelnen Karten und den verschiedenen Programmen? Wenn Sie umfassend informiert sein wollen, finden Sie in diesem Buch die Antworten. Dabei erfahren Sie auch gleich, wie Sie Zeichensätze und Grafiken editieren, Hardcopy- und Snapshot-Routinen erstellen und nutzen, Dia-Shows und Animationen programmieren und und und.
**Das große PC-Grafikbuch**
**Hardcover, inkl. Diskette, ca. 400 Seiten, DM 59,–**
**erscheint ca. 8/88**




**dBase III Plus – Tips & Tricks**
Allen Anwendern, die bereits mit dBase III Plus arbeiten, gibt dieses Buch den letzten Schliff. Hier finden Sie alles, was die Arbeit mit diesem ausgefeilten Datenbank-System richtig professionell gestaltet: Merkmale der Punktoberfläche, Variablenbehandlung, Datenverwaltung, Programmoptimierung, Daten- und Programmübernahme aus früheren Versionen, Datenaustausch und als besonderes Bonbon einen kleinen Ausblick auf das, was das neueste dBase IV von Ashton Tate zu bieten hat. Dazu jede Menge Hilfstoutinen und zahlreiche Tips zur praktischen Arbeit. Besonders hilfreich: die Diskette zum Buch mit allen Beispielen.
**Tips & Tricks für dBase III Plus-Anwender**
**Hardcover, 444 Seiten, inkl. Diskette, DM 69,–**




**Word für Einsteiger**
Word starten und gleich die ersten Texte schreiben. Mit Word für Einsteiger. Bereits nach einem Abend beherrschen Sie die wichtigsten Grundlagen dieser komplexen Textverarbeitung. Textformatierung, Serienbriefe und das Ausdrucken bleiben keine Geheimnisse mehr. Neben den Word-Versionen 2, 3 und Junior werden auch die erheblichen Erweiterungen der Word-Version 4 erklärt: Datei-Manager zur Dokumentenverwaltung, neue Funktionen zur Textüberarbeitung und die Menüführung.
**Word für Einsteiger**
**Hardcover, 267 Seiten, DM 49,–**




**oftware-Seminar MS-DOS**
**92 Seiten, DM 79,–**
**ahlweise mit 5 ¼"- oder 3 ½"-Diskette**

## Schnell-Lehrgang für Erfolgsverwöhnte.

Der PC ist für Sie in erster Linie eins: ein überaus praktisches Hilfsmittel, um mit der sich täglich ansammelnden Arbeit besser und schneller als bisher fertig zu werden. Wie es in diesem Rechner so ausschaut, interessiert Sie dabei herzlich wenig. Hauptsache, er macht genau das, was Sie wollen. Doch das braucht seine Zeit. Und Zeit ist Geld. Im Software-Seminar MS-DOS finden Sie daher eine gezielte Unterweisung zu allen elementaren Vorgängen wie Kopieren, Formatieren, Pfade erstellen, Batch-Dateien und und und. Immer auf das Wesentliche beschränkt. So haben Sie bereits innerhalb kürzester Zeit das Betriebssystem und damit Ihren neuen PC im Griff – und verfügen auch schon über die nötige praktische Erfahrung. Dank einer beigefügten Trainingsdiskette (wahlweise 5 ¼" oder 3 ½") mit zahlreichen Übungsbeispielen. Hier finden Sie dann auch gleich eine Reihe lauffähiger Tools und fertiger Anwendungen, wie z. B.
ein Benutzerlogbuch, eine Batch-Datei zur automatischen Dateisicherung, einen Dateisucher, ein Notizbuch mit Terminverwaltung und und und. Schneller führt kaum ein Weg zum praktischen Umgang mit MS-DOS.

**COUPON**

COUPON BITTE EINSENDEN AN:
DATA BECKER
MEROWINGERSTR. 30
4000 DÜSSELDORF

Zzgl. DM 5,– Versandkosten unabhängig von der best. Stückzahl

HIERMIT BESTELLE ICH FÜR MEINEN PC

NAME, VORNAME

STRASSE

ORT

☐ BITTE SENDEN SIE MIR KOSTENLOS UND UNVERBINDLICH IHREN NEUEN KATALOG.

MEIN COMPUTER: _____

# Einblick in den Speicher – residente Programme aufgespürt

**Speicherresidente Programme sind eine feine Sache. Ob MS-DOS-Hilfe auf Tastendruck oder Taschenrechner und Jahrhundertkalender mit Terminplan, auf diese flinken Helfer möchte man ungern verzichten. Unser Utility »tsrmap« zeigt Ihnen, an welcher Stelle in Ihrem Speicher sich diese residenten Programme befinden, und vieles mehr.**

Seit PCs allgemein mit großem Speicher ausgerüstet sind, bereitet es keine Probleme mehr, mehrere speicherresidente Programme gleichzeitig darin zu halten. Selbst Programme, von denen Sie kaum Notiz nehmen, zum Beispiel »command.com« oder »keybgr.com«, sind ständig im Speicher präsent, ohne jedesmal von der Diskette oder Festplatte neu geladen zu werden. »tsrmap« (Listing) bringt Licht in das Dunkel Ihres Speichers.

Das Programm verschafft Ihnen die nötige Übersicht über alle augenblicklich aktiven Programme in Ihrem Speicher – mit Ausnahme der Gerätetreiber (»Device Drivers«). Gerade beim Einsatz mehrerer TSR-Programme (»Terminate and Stay Resident«) ergeben sich oft Schwierigkeiten, wenn sich zwei dieser Programme untereinander nicht vertragen. Meist findet das Gerangel in der Interrupt-Tabelle statt, die nur einen Vektor pro Interrupt aufnehmen kann.

Wenn Sie sich selbst an das Entwerfen dieser Art von Programmen wagen (für Sie seit dem Kurs in der DOS International eine Selbstverständlichkeit), gibt Ihnen »tsrmap« Auskunft über Ihre Fortschritte.

Zunächst einige Erläuterungen zu Begriffen und Abkürzungen, die Ihnen im Rahmen dieses Artikels immer wieder begegnen werden:

»TSR« – Terminate and Stay Resident
Ein geladenes Programm wird formal beendet, bleibt aber im Speicher erhalten (belegt Speicherplatz und benutzt in der Regel auch Interrupt-Vektoren, die in das Programm zeigen).

»PSP« – Program Segment Prefix
Ein Programmvorspann, der von MS-DOS (PC-DOS) vor dem Laden eines Programms erzeugt wird und nach dem Laden zum Programm gehört. Bei speicherresidenten Programmen bleibt dieser Vorspann ebenfalls erhalten.

»ENV« – Environment Block
Die Programmumgebung stellt einen Kommunikationsbereich zwischen den einzelnen Programmen dar (siehe auch den MS-DOS-Befehl »set«). Bei speicherresidenten Programmen kann dieser Bereich entfernt worden sein.

»MCB« – Memory Control Block
Ein 16 Byte großer Speicherbereich, der Informationen über einen Ausschnitt des Speichers enthält.

Selbstverständlich gilt das für MS-DOS Erwähnte ebenso für PC-DOS, denn die Unterschiede zwischen diesen beiden Versionen haben auf »tsrmap« keinen Einfluß.

»tsrmap« gibt über jedes im Speicher befindliche TSR-Programm folgende Informationen aus (Bild):
- Adr.: Segment-Adresse des PSP
- Grösse: Gesamtanzahl (dezimal) der reservierten Bytes
- Name: vollständiger Programmname mit Laufwerk, Pfad und Erweiterung (nur wenn ENV noch vorhanden)

```
A:\>tsrmap

TSRMAP - Anzeige der speicherresidenten Programme        (c) 1988 Manfred Mager

Adr.  Grösse  Name                                          Interrupts
-----------------------------------------------------------------------
10A9    3088  C:\WORD;C:\FW3;C:\;D:\DBASEIII                22 23 24 2E
11A1    3680  C:\SYS\DOS\CKEYGR.COM                         09
168A     432  C:\SYS\ZLUPE\ZLB.COM                          1C
-----------------------------------------------------------------------
noch  579056  Bytes frei

A:\>_
```

**So werden die Informationen dargestellt**

- Interrupts: Interrupt-Nummern (hexadezimal), die in das Programm zeigen, optional auch mit vollständigem Vektor (Segment:Offset)

Die Zeichenausgabe erfolgt immer über den MS-DOS-Interrupt 21h (Funktionen 02hex – zeichenweise, oder 09hex – String bis »$«) auf die Standard-Ausgabeeinheit. Das heißt, Sie können Zeichen auch ausdrucken oder in eine Datei ausgeben. Beim Aufruf von »tsrmap« kann ein Schalter (»switch«) als Parameter übergeben werden. Erlaubte Parameter sind:
- ?: Programm-Info und Anzeige der möglichen Parameter
- /v oder /V: Schalter für Anzeige der Interrupt-Vektoren

Als Schaltereinleitung wird sowohl der Schrägstrich »/« (MS-DOS) wie auch das Minuszeichen » – « (Unix) erkannt. Groß- und Kleinbuchstaben werden gleich behandelt.

Das Programm ist in 8088-Assembler geschrieben und auf allen PC-Versionen unter MS-DOS (PC-DOS) ab Version 2.xx lauffähig. Es ist als Com-Programm konzipiert und etwas über 1 KByte groß (allein die Textkonstanten belegen schon etwa 300 Byte).

## »tsrmap« gibt über jedes speicherresidente Programm Informationen aus

Nach der Eröffnung des Segments »Code« an der Adresse 0100hex steht der Sprung zum Hauptteil. Danach folgen die Texte für »tsrmap«:
- Übersicht der speicherresidenten Programme
- Programm-Info
- Überschriften und Hilfsanzeigen.

Ein paar Bytes sind anschließend für diverse Variablen reserviert. Das erste Unterprogramm »part01« initialisiert zwei Variablen und wertet die Befehlszeile aus. Wird ein Fragezeichen aufgefunden, so zeigt das Programm die Überschrift und den Hilfetext an. Das Carry-Flag wird gesetzt, damit der Hauptteil anschließend direkt zum Ende verzweigt. Die Auswertung der Befehlszeile ist denkbar einfach und für Erweiterungen gut geeignet. Für jeden Schalter ist hier ein Byte verwendet, was bei komplexen Parameter-Kombinationen sicherlich sehr hilfreich ist. In der aktuellen Version von »tsrmap« gibt es nur einen Schalter, dessen Zustand am Feld »memory_switch« zu erkennen ist.

Zurück im Hauptteil, folgt der Aufruf von »part02«, der wichtigsten Funktion des Programms, von deren korrekter Ausführung alles weitere abhängig ist: das Auffinden des ersten PSP im Speicher.

An dieser Stelle bietet es sich an, die Speicherorganisation von MS-DOS im PC näher zu beschreiben. Wie in früheren Ausgaben von DOS International schon erläutert, läßt sich der Speicheranfang wie folgt vereinfacht aufteilen:

- 0000:0000 bis 0000:03ff Interrupt-Vektoren
- 0040:0000 bis 0060:00ff DOS/Basic/div.-Variablenspeicher
- 0070:0000 bis ????:???? von hier ab werden die Gerätetreiber beim Booten von MS-DOS abgelegt

Da die Größe des benötigten Speichers für Gerätetreiber (»Device Drivers«) von der MS-DOS-Version und der Anzahl der geladenen Treiber abhängig ist, kann die Adresse des ersten residenten Programms (»command.com«) nur experimentell ermittelt werden – auf jeden Fall ist sie größer als 0070:0000. Aus diesem Grund wird das Register AX mit dem Wert 0070hex vorbesetzt.

Wie erkennt man aber den Anfang eines Programms im Speicher? Jedem Programm (»com« und »exe«) geht ein PSP voraus, der auch bei residenten Programmen noch erhalten ist. Da jeder PSP glücklicherweise eindeutige Kennzeichen trägt, lassen sich diese auch relativ sicher zur PSP-Bestimmung verwenden. Das Ihnen vorliegende Programm benutzt den Aufruf »int 20hex« bei Adresse PSP:0000 als ersten Vergleich. Scheitert dieser Vergleich, wird sofort weitergesucht (wenn es sein muß bis zum RAM-Ende 9fff:0000). War der erste Vergleich schon erfolgreich, so ist noch lange nicht gesagt, daß dieser »int 20hex« nicht das Ende einer Programmausführung bedeutet oder einfach ein Überbleibsel eines längst verarbeiteten Programms ist. Aus diesem Grund müssen noch weitere Tests folgen.

## Mit »tsrmap« können Sie selbstgeschriebene TSR-Programme kontrollieren

In jedem PSP seit der MS-DOS-Version 2.xx steht bei Adresse PSP:0050 ein »int 21hex«. Funktioniert dieser Vergleich auch, so sind Sie dem ersten PSP schon ziemlich nahe. Und trotzdem – Aufrufe »int 21hex« gibt es in MS-DOS-Programmen geradezu wie Sand am Meer. Deshalb wird nun eine Speicherstelle abgefragt, die genau 16 Byte (1 Paragraph) vor dem PSP liegt: (PSP-1):0000. Das wäre die Adresse des zugehörigen Speicherverwaltungsblocks »MCB«. Jeder MCB trägt in seinem ersten Byte ein »M« (oder ein »Z«, wenn es sich um den letzten MCB in der Kette handelt). Genau dieses Kennzeichen wird nun überprüft. Dieser letzte Vergleich sollte nun absolut sicherstellen, daß kein TSR-Programm mehr entschlüpfen kann: nach dem »M« muß nun auch noch die Adresse des eben gefundenen PSP stehen. So kompliziert es auch klingt, diese vier Vergleiche gewährleisten, daß Sie nun die Adresse des ersten PSP sowie dessen MCB gefunden haben.

Der Aufbau eines Speicherverwaltungsblocks (MCB) noch einmal in Kürze: An der Adresse MCB:0000hex befindet sich immer die Kennzeichnung »M« oder »Z«. Die nächsten zwei Bytes ab MCB:0001hex zeigen auf den Anfang des verwalteten Speicherbereichs. Danach folgen zwei weitere Bytes, die die Größe dieses Speicherbereichs in Paragraphen (also 16 Byte) enthalten (MCB:0003hex). Addiert man nun diese Größe zur Anfangsadresse des MCB dazu und berücksichtigt, daß der MCB selbst 16 Byte groß ist, so erhält man die Adresse des nächsten MCB. Diese Kette läßt sich nun bis zum letzten MCB (markiert mit »Z«) fortsetzen, der als Größenangabe den Rest in Paragraphen bis zum RAM-Ende enthält.

Die in »part02« angewendete Methode funktioniert nur, wenn das erste residente Programm keine eigene Programmumgebung (ENV) hat. Die Begründung hierfür ist eigentlich ganz einfach: Beim Laden eines Programms kopiert der Eltern-Prozeß (»Parent Process«) sein eigenes ENV, erzeugt einen neuen PSP und lädt erst dann das Programm in den Speicher. Aus dieser Reihenfolge wird klar, daß der erste freie Speicherbereich dem ENV, und frühestens der nächste freie Speicherblock dem Programm mit seinem PSP zugeordnet ist. Da ein ENV immer kleiner ist als das dazugehörige Programm plus PSP, wird diese Reihenfolge auch immer eingehalten.

Die Tatsache, daß die Adresse in MCB:0001hex immer auf den PSP zeigt, ermöglicht es, die Größen von ENV- und Programmbereich zu addieren und so die tatsächliche Anzahl reservierter Bytes für ein TSR-Programm zu ermitteln. In »part03« des Programms »tsrmap« wird deshalb die PSP-Adresse in »last_segm« für spätere Vergleiche zwischengespeichert. Erst beim Wechsel dieser Adresse wird eine Programm-Info von »part04« ausgegeben. Beim Rest von »part03« handelt es sich um die Ermittlung der nächsten MCB-Adresse sowie um eine Überprüfung, ob es sich bei der neuen Adresse noch um einen MCB handelt. In diesem Fall geht die Suche wie gewohnt weiter. Da das Carry-Flag nach der Berechnung der nächsten Speicheradresse immer gelöscht ist, wird es eingesetzt, um der Hauptschleife mitzuteilen, daß das Ende der MCB-Kette erreicht ist.

## Sie finden alle TSR-Informationen in einer Zeile

Die Ausgabe einer Programm-Info in »part04« bereitet das Programm durch Konvertierungen der Ausgabefelder »PSP-Segment« und »Größe« ins ASCII- oder Dezimalformat vor. Anschließend beginnt die Suche nach dem Programmnamen – eine nicht ganz einfache Sache. Die Reihenfolge, in der MS-DOS Programme in den Speicher lädt, ist bereits beschrieben worden. Nachzutragen bleibt noch die Beschreibung über das Auffinden eines Programmnamens im Speicher.

Direkt im Anschluß an die ENV-Angaben findet sich der Wert »0000hex« oder »0001hex«. Beträgt dieser Wert »0000hex«, so handelt es sich um das erste Auftreten von »command.com« im Speicher. In diesem Fall setzt das Programm den Zeiger auf den Anfang des ENV zurück und zeigt den Inhalt der Comspec-Umgebungsvariablen an.

In allen anderen Fällen, wenn »0001hex« an dieser Stelle zu finden ist, folgt der vollständige Programmname mit Laufwerks- und Pfadangabe. Diesem glücklichen Umstand ist zu verdanken, daß ein Programm wie »tsrmap« überhaupt sinnvolle Angaben über TSR-Programme machen kann.

Problematischer ist es, wenn der Speicherbereich, den einmal das ENV eines TSR-Programms belegt hat, vom Programmierer aus Sparsamkeit freigegeben wurde. In diesem Fall haben Sie keine Chance mehr, den Programmnamen im Speicher zu finden. »tsrmap« zeigt dann auf dem Bildschirm »? ? ?« an.

Ein weiterer unschöner Effekt dieser Arbeitsweise liegt darin, daß eventuell die ENV- und PSP-Bereiche des nächsten Programms nicht mehr direkt aufeinanderfolgen. »tsrmap« liest alle MCBs sequentiell und gibt bei Wechsel des PSP eine Programm-Info aus. Das heißt, man erhält für ein TSR-Programm zwei Ausgaben, die eine für das ENV, die andere für den PSP, beide mit ihrer jeweiligen Größenangabe, wenn die MCBs für ein Programm nicht sequentiell angeordnet sind.

Nach der Anzeige von PSP-Adresse, Programmgröße und Programmnamen folgt eine weitere wichtige Information über das TSR-Programm: die verwendeten Interrupts. Falls Sie beim Aufruf von »tsrmap« den Schalter »/v« eingesetzt haben, erhalten Sie hier außerdem den aktuellen Interrupt-Vektor (Segment:Offset). Der Programmteil

»part05« ist eine Schleife, die alle Interrupt-Vektoren von 00hex bis ffhex unter dem Gesichtspunkt prüft, ob sie in den reservierten Speicherbereich des gerade angezeigten Programms zeigen. Ist dies der Fall, zeigt es die Interrupt-Nummer (und gegebenenfalls die Vektor-Adresse) an.

## Die Speicherverwaltungsblöcke werden mehrmals abgefragt

Dieser Programmteil arbeitet sehr langsam, da er aus Kompatibilitätsgründen die Funktion 35hex des MS-DOS-Interrupts 21hex benutzt und jeden Interrupt von neuem testet, egal ob er vorher bereits angezeigt wurde oder nicht.

Nachdem nun alle speicherresidenten Programme bekannt sind, gibt das Programm noch die Anzahl der freien Bytes im MS-DOS-Hauptspeicher (ohne Extended Memory) aus. Bei der Ermittlung des freien Speichers wird man jedoch mit einer Besonderheit konfrontiert, die im Vorfeld noch nicht aufgefallen war: verschiedene Programme zeigen den freien Speicher an, und jedes von ihnen kommt zu einem anderen Ergebnis. Des weiteren muß die Annahme, daß das Programm »chkdsk« von MS-DOS eine korrekte Angabe über die zur Verfügung stehenden Bytes liefert, schon bald korrigiert werden. Ausgiebige Tests ergaben, daß »chkdsk« einmal die Anzahl Bytes allein aus dem letzten MCB anzeigt, manchmal zwei, hin und wieder drei Paragraphen abzieht, aber grundsätzlich seine eigene ENV-Größe bei der Ausgabe vergißt.

Der restliche freie Speicher ergibt sich aus der Summe der Größenangaben der MCBs, die auf »tsrmap« weisen, plus den Größen der Bereiche, deren MCBs auf 0000hex zeigen. Es interessiert jedoch der Umfang des tatsächlich freien Bereichs und nicht derjenige, der dem Anwendungsprogramm (zum Beispiel »chkdsk« oder »tsrmap«) zur Verfügung steht. Wenn Sie die Applikation »tsrmap« speicherresident machen, müssen Sie diese Berechnungsart anpassen.

Sie verfügen mit »tsrmap« über ein Programm, das Ihnen eine nützliche Hilfe ist. Selbst wenn Sie »nur« Routinen in Ihre eigene Software übernehmen, hat dieses Programm seinen Zweck schon erfüllt.

(Manfred Mager/gh)

```
[CV2]    0:                page      62,132
[YMD]    1:                title     TSRMAP.ASM - displa
           y resident programs
[000]    2:
[067]    3: code           segment   para 'CODE'
[GE7]    4:                assume    cs:code, ds:code
[KFB]    5:                org       0100h
                ;'.COM' Startadresse
[000]    6:
[W4F]    7: tsrmap:        jmp       anfang
                ;zum Programmanfang
[000]    8:
[UXD]    9: header_line    db        0dh,0ah
                ;Überschrift
[3BH]   10:                db        'TSRMAP - Anzeige d
           er speicherresidenten Programme'
[UU3]   11:                db        8 dup (32)
[W27]   12:                db        '(c) 1988 Manfred M
           ager'
[GW2]   13:                db        0dh,0ah
[114]   14:                db        79 dup (205)
[GW2]   15:                db        0dh,0ah
[1S6]   16: header_end     label     byte
[1M7]   17:                db        'Adr.  Grösse  Name
                                     '
[ER3]   18:                db        39 dup (32)
[7C5]   19:                db        'Interrupts'
[GW2]   20:                db        0dh,0ah
[ZR6]   21: underline      label     byte
[YO4]   22:                db        04 dup (196)
[7T1]   23:                db        ' '
[614]   24:                db        06 dup (196)
[7T1]   25:                db        ' '
[YO4]   26:                db        41 dup (196)
[7T1]   27:                db        ' '
[WO4]   28:                db        22 dup (196)
[GD3]   29:                db        0dh,0ah,'$'
[000]   30:
[V5H]   31: help_text      db        'Aufruf : TSRMAP  [
           /switch] [-switch] [?]'
[404]   32:                db        0dh,0ah,0ah
[NMD]   33:                db        ' /v -V : Anzeige d
           er Interrupt-Vektoren'
[8H7]   34: crlf_char      db        0dh,0ah,'$'
[000]   35:
[9WF]   36: prgm_addr      db        '0000 '
                ;Hilfsfelder für Anzeige
[TD6]   37: prgm_size      db        '      $'
[YE6]   38: interrupt      db        '  00$'
[HT6]   39: vector         db        '  0000:0000$'
[AUD]   40: unbekannt      db        '? ? ?',00h
                ;Programm-Name ?
[PO7]   41: free_mem_1     db        'noch $'
[W78]   42: free_mem_2     db        'Bytes frei'
[GD3]   43:                db        0dh,0ah,'$'
[000]   44:
[GSD]   45: tsrmap_psp     dw        ?
                ;eigene PSP-Adresse
[ASD]   46: tsrmap_env     dw        ?
                ;eigene ENV-Adresse
[15B]   47: memory_size    dw        ?
                ;RAM-Grösse
[W9H]   48: vector_switch  db        ?
                ;Vektor-Anzeige Schalter
[000]   49:
```

```
[43E]   50: reg_bx         dd        ?
                ;16-Bit Hilfsspeicher
[HHD]   51: last_segm      dw        ?
                ;letztes Segment
[MLD]   52: segm_addr      dw        ?
                ;aktuelles Segment
[TLD]   53: segm_size      dw        ?
                ;Programm-Grösse
[3PE]   54: segm_psp       dw        ?
                ;Programm PSP-Adresse
[1SD]   55: vector_segm    dw        ?
                ;Vektor-Segment
[HUD]   56: vector_offs    dw        ?
                ;Vektor-Offset
[DOE]   57: counter        db        ?
                ;Zähler für Ausgabe
[000]   58:
[VZO]   59:                page
[7BE]   60: anfang:        call      part01
                ;Initialisierung
[1X9]   61:                jc        ok
                ;nur Hilfszeile
[000]   62:
[4L9]   63:                call      part02
                ;1. PSP suchen
[L89]   64:                inc       al
                ;Fehler-Code = 1
[4HA]   65:                jc        ende
                ;nichts gefunden
[000]   66:
[7PH]   67: naechster:     mov       word ptr [reg_bx],b
          x      ;PSP-Adresse retten
[OP8]   68:                mov       word ptr [reg_bx+02
           h],es
[KFD]   69:                call      part03
                ;Programm-Info anzeigen
[9HF]   70:                jnc       naechster
                ;weiter, wenn nicht letztes
[000]   71:
[HAE]   72:                call      part06
                ;Anz. freie Bytes ausgeben
[XWA]   73: ok:            xor       al,al
                ;kein Fehler
[2ED]   74: ende:          mov       ah,4ch
                ;Programm beenden
[392]   75:                int       21h
[000]   76:
[VAB]   77: part01:        mov       ax,cs
                ;TSRMAP - PSP
[G9A]   78:                mov       tsrmap_psp,ax
                ;merken
[2U3]   79:                xor       bx,bx
[UM9]   80:                mov       ax,[bx+2ch]
                ;TSRMAP - ENV
[39A]   81:                mov       tsrmap_env,ax
                ;merken
[N8B]   82:                cld
                ;aufsteigend zählen
[1YB]   83:                mov       si,81h
                ;Beginn der Parameter
[78D]   84: part01_a:      lodsb
                ;Zeichen einlesen
[1KC]   85: part01_b:      or        al,al
                ;Zeichen == '\0' ?
[VS3]   86:                jz        part01_d
```

```
[000]   87:
[6P9]   88:                     cmp         al,0dh
                ;Zeichen == '\n' ?
[HR3]   89:                     je          part01_d
[000]   90:
[EW8]   91:                     cmp         al,'/'
                ;Zeichen == '/' ?
[FR3]   92:                     je          part01_c
[000]   93:
[YV8]   94:                     cmp         al,'-'
                ;Zeichen == '-' ?
[FR3]   95:                     je          part01_c
[000]   96:
[E09]   97:                     cmp         al,'?'
                ;Zeichen == '?' ?
[CY3]   98:                     jne         part01_a
[000]   99:
[YK5]  100:                     mov         header_end,'$'
[5EH]  101:                     mov         dx,offset header_li
        ne      ;Überschrift anzeigen
[NE4]  102:                     call        display
[L7F]  103:                     mov         dx,offset help_text
                ;Hilfetext anzeigen
[NE4]  104:                     call        display
[4M0]  105:                     stc
[XD5]  106:                     jmp short   part01_d
[000]  107:
[05F]  108: part01_c:           lodsb
                ;Switch-Zeichen lesen
[ESB]  109:                     or          al,20h
                ;==> Kleinbuchstaben
[B8A]  110:                     cmp         al,'v'
                ;Zeichen == 'v'/'V' ?
[CY3]  111:                     jne         part01_a
[000]  112:
[90G]  113:                     mov         vector_switch,1
                ;Vektor-Adresse ausgeben
[7WA]  114:                     jmp         part01_a
                ;weiterlesen
[000]  115:
[W8F]  116: part01_d:           ret
                ;Ende der Initialisierung
[000]  117:
[C6A]  118: part02:             push        ds
                ;DS retten
[T59]  119:                     mov         dx,offset header_li
        ne
[1EE]  120:                     call        display
                ;Überschrift anzeigen
[85B]  121:                     mov         ax,0070h
                ;1. Möglichkeit im RAM
[9YD]  122:                     xor         bx,bx
                ;Offset für Speicher-Test
[90D]  123: part02_a:           cmp         ax,tsrmap_psp
                ;== TSRMAP ?
[DR3]  124:                     je          part02_b
[000]  125:
[FT3]  126:                     mov         ds,ax
[5WA]  127:                     cmp         word ptr [bx],20cdh
                ;== int 20h ?
[EY3]  128:                     jne         part02_b
[000]  129:
[XSA]  130:                     cmp         word ptr [bx+50h],2
        1cdh    ;== int 21h ?
[EY3]  131:                     jne         part02_b
[000]  132:
[10D]  133:                     mov         dx,ax
                ;Adresse von Memory-
[B19]  134:                     dec         dx
                ;   Control-Block
[4U3]  135:                     mov         es,dx
[9M9]  136:                     cmp         byte ptr es:[bx],4d
        h       ;== 'M' ?
[EY3]  137:                     jne         part02_b
[000]  138:
[6GC]  139:                     cmp         ax,word ptr es:[bx+
        01h]    ;== PSP-Adresse ?
[EY3]  140:                     jne         part02_b
[000]  141:
[RAA]  142:                     clc
                ;Carry-Flag löschen
[AKD]  143:                     jmp short   part02_c
                ;ok - 1. PSP gefunden
[000]  144:
[E1B]  145: part02_b:           inc         ax
                ;Segment =+ 1
[FL9]  146:                     cmp         ah,0a0h
                ;noch DOS-Ram ?
[YY6]  147:                     jb          part02_a
                ;ok
[000]  148:
[EPA]  149:                     stc
                ;Carry-Flag f. Fehler
[H9E]  150: part02_c:           pop         ds
                ;DS wiederherstellen
[TVC]  151:                     ret
                ;zurück zum Hauptprogramm
[000]  152:
[S1H]  153: part03:             mov         dx,es:[bx+01h]
                ;Adresse aus Memory-Block
```

```
[FPA]  154:                     or          dx,dx
                ;== letzter Block ?
[FR3]  155:                     je          part03_c
[000]  156:
[E8H]  157:                     cmp         dx,last_segm
                ;== gleiches Programm wie voher ?
[DR3]  158:                     je          part03_b
[000]  159:
[NRA]  160:                     cmp         dx,tsrmap_psp
                ;== TSRMAP ?
[FR3]  161:                     je          part03_c
[000]  162:
[APA]  163:                     cmp         dx,tsrmap_env
                ;== TSRMAP ?
[FR3]  164:                     je          part03_c
[000]  165:
[7CC]  166:                     cmp         last_segm,0
                ;== erstes Programm ?
[77C]  167:                     mov         last_segm,dx
                ;Adresse retten
[BR3]  168:                     je          part03_a
[000]  169:
[VED]  170:                     call        part04
                ;Programm-Info ausgeben
[AFL]  171: part03_a:           mov         segm_addr,dx
                ;PSP-Adresse für Ausgabe merken
[4U3]  172:                     mov         es,dx
[2U3]  173:                     xor         bx,bx
[D9D]  174:                     mov         dx,es:[bx+2ch]
                ;ENV-Adresse holen
[CVD]  175:                     mov         segm_psp,dx
                ;  und für Ausgabe merken
[MCL]  176:                     les         bx,dword ptr [reg_b
        x]      ;letzte Memory-Block Adresse
[KYM]  177: part03_b:           mov         ax,es:[bx+03h]
                ;Grösse des verwalteten Blocks
[K5A]  178:                     add         segm_size,ax
                ;addieren
[XD5]  179:                     jmp short   part03_d
[000]  180:
[MBH]  181: part03_c:           mov         ax,es:[bx+03h]
                ;Grösse von TSRMAP-Block
[6TC]  182:                     add         memory_size,ax
                ;Gesamtspeicher
[9M9]  183:                     cmp         byte ptr es:[bx],4d
        h       ;== 'M' ?
[S7K]  184:                     jne         part04
                ;Ende, letzte Programm-Info ausgeben
[000]  185:
[1ZF]  186:                     mov         ax,es:[bx+03h]
                ;Grösse Memory Block holen
[DM9]  187: part03_d:           cmp         ax,1000h
                ;> 64k ?
[DR3]  188:                     jb          part03_e
[000]  189:
[BBE]  190:                     mov         dx,ax
                ;Segment-Register anpassen
[US3]  191:                     and         ax,0f000h
[TS3]  192:                     sub         dx,ax
[ZF2]  193:                     push        dx
[ST3]  194:                     mov         dx,es
[BR3]  195:                     add         dx,ax
[4U3]  196:                     mov         es,dx
[162]  197:                     pop         ax
[G1C]  198: part03_e:           mov         cl,04h
                ;Grösse =* 16
[CP3]  199:                     shl         ax,cl
[GG8]  200:                     add         ax,10h
                ;Grösse =+ 16
[HDE]  201:                     add         bx,ax
                ;Offset-Register anpasssen
[DUA]  202:                     jnc         part03_f
                ;kein Überlauf
[000]  203:
[RAE]  204:                     mov         ax,es
                ;Segment-Register anpassen
[A63]  205:                     add         ax,1000h
[KT3]  206:                     mov         es,ax
[YRF]  207: part03_f:           ret
                ;zurück zur Hauptschleife
[000]  208:
[ROD]  209: part04:             push        dx
                ;Register retten
[BF2]  210:                     push        ds
[T52]  211:                     pop         es
[5TB]  212:                     mov         dx,segm_addr
                ;PSP Segment
[UCD]  213:                     mov         di,offset prgm_addr
                ;in ASCII wandeln
[YZ4]  214:                     call        conv2ascii
[UDC]  215:                     mov         ax,segm_size
                ;Programm-Grösse
[RND]  216:                     mov         di,offset prgm_size
        +6      ;letzte Stelle
[AHD]  217:                     mov         si,offset prgm_size
                ;erste Stelle
[2RD]  218:                     call        conv2dec
                ;in Dezimalzahl wandeln
[B4E]  219:                     mov         dx,segm_psp
                ;Programm-Namen suchen
```

```
[MT6]  220:                cmp       dx,tsrmap_env
[DR3]  221:                je        part04_b
[000]  222:
[PYD]  223:                mov       es,dx
                     ;Segment des Programm ENV
[NDD]  224:                mov       di,0ffffh
                     ;Offset für Such-Schleife
[BX4]  225: part04_a:      inc       di
[GWE]  226:                cmp       byte ptr es:[di],00
           h       ;== Ende eines Eintrags ?
[OGD]  227:                jnz       part04_a
                     ;nein, weiter suchen
[000]  228:
[Y22]  229:                inc       di
[3VE]  230:                cmp       byte ptr es:[di],00
           h       ;== Ende aller Einträge ?
[OGD]  231:                jnz       part04_a
                     ;nein, weiter suchen
[000]  232:
[Y22]  233:                inc       di
[DZC]  234:                cmp       word ptr es:[di],00
           01h     ;== '0001' ? Konstante
[HR3]  235:                je        part04_d
[000]  236:
[2ZC]  237:                cmp       word ptr es:[di],00
           00h     ;== '0000' ? Konstante
[MYA]  238:                mov       di,0ffffh
                     ;neuer Offset = 0
[TS3]  239:                jz        part04_c
[000]  240:
[Z3F]  241: part04_b:      mov       di,offset unbekannt
           - 2     ;'? ? ?' - Text
[BF2]  242:                push      ds
[T52]  243:                pop       es
[XD5]  244:                jmp short part04_d
[000]  245:
[BX4]  246: part04_c:      inc       di
[NSG]  247:                cmp       byte ptr es:[di],00
           h       ;Ende des Eintrags erreicht ?
[DR3]  248:                je        part04_b
[000]  249:
[Y3C]  250:                cmp       byte ptr es:[di],'=
           '       ;für COMSPEC= ...
[GY3]  251:                jne       part04_c
[000]  252:
[722]  253:                dec       di
[CAK]  254: part04_d:      mov       dx,offset prgm_addr
                     ;Programm-Info Teil 1
[NE4]  255:                call      display
[CKC]  256:                mov       cx,41
                     ;41 Bytes maximal anzeigen
[Y22]  257:                inc       di
[BSC]  258:                mov       ah,02h
                     ;Zeichenweise ausgeben
[BX4]  259: part04_e:      inc       di
[NED]  260:                mov       dl,es:[di]
                     ;Programm-Namen lesen
[RE3]  261:                or        dl,dl
[7VA]  262:                jz        part04_f
                     ;Ende des Namens ?
[000]  263:
[G9B]  264:                int       21h
                     ;Programm-Info Teil 2
[P94]  265:                loop      part04_e
[000]  266:
[UOE]  267: part04_f:      or        cx,cx
                     ;41 bytes erreicht ?
[3T3]  268:                jz        part04_h
[000]  269:
[5GA]  270:                mov       dl,' '
                     ;mit Leerstellen
[80B]  271: part04_g:      int       21h
                     ;auffüllen
[U94]  272:                loop      part04_g
[000]  273:
[1RH]  274: part04_h:      call      part05
                     ;Interrupt-Vektoren anzeigen
[EZA]  275:                call      crlf
                     ;Zeilenvorschub
[3UD]  276:                pop       dx
                     ;Register wiederherstellen
[97B]  277:                stc
                     ;Flag für Ende setzen

[SMO]  278:                ret
[000]  279:
[LT5]  280: part05:        xor       al,al
[YAD]  281:                mov       counter,al
                     ;Zähler für Anzeige
[8E8]  282: part05_a:      mov       cx,segm_addr
[ZCC]  283:                mov       ah,35h
                     ;Interrupt-Vektor lesen
[392]  284:                int       21h
[Y7B]  285:                mov       dx,es
                     ;Segment-Adressen
[DSA]  286:                cmp       cx,dx
                     ;vergleichen
[1GA]  287:                ja        part05_f
                     ;passt nicht
```

```
[000]  288:
[V3F]  289:                mov       vector_segm,dx
                     ;Vektor-Segment merken
[Z1F]  290:                mov       vector_offs,bx
                     ;Vektor-Offset merken
[HYE]  291:                add       cx,segm_size
                     ;prüfen, ob Vektor-Offset
[WXC]  292:                shr       bx,1
                     ;noch im Programmbereich
[1P2]  293:                shr       bx,1
[958]  294:                shr       bx,1
                     ;Offset =/ 16
[1P2]  295:                shr       bx,1
[HR3]  296:                add       dx,bx
[KT3]  297:                cmp       cx,dx
[1S9]  298:                jb        part05_f
                     ;doch nicht
[000]  299:
[X45]  300: part05_b:      push      ax
[T3C]  301:                cmp       counter,6
                     ;schon 6 Vektoren ?
[KY3]  302:                jne       part05_d
[000]  303:
[EZA]  304:                call      crlf
                     ;Zeilenvorschub
[AZ2]  305:                mov       ah,02h
[YB7]  306:                mov       cx,55
                     ;Anzahl
[EK9]  307:                mov       dl,' '
                     ;Leerstellen
[ZPE]  308: part05_c:      int       21h
                     ;bis Spalte 'Interrupt'
[PA9]  309:                loop      part05_c
                     ;ausgeben
[000]  310:
[TY9]  311:                mov       counter,cl
                     ;Zähler = 0
[162]  312:                pop       ax
[PF2]  313:                push      ax
[ATC]  314: part05_d:      inc       counter
                     ;Zähler =+ 1
[KC8]  315:                mov       cx,2
                     ;nur 8 Bit
[FN3]  316:                mov       dh,al
[LHE]  317:                mov       di,offset interrupt
           + 2     ;ab hier speichern
[TLD]  318:                call      conv2ascii_a
                     ;Zeichen umwandeln
[S58]  319:                mov       dx,offset interrupt
[P2D]  320:                call      display
                     ;Interrupt anzeigen
[1DG]  321:                cmp       vector_switch,0
                     ;Vektor-Adresse anzeigen ?
[CP7]  322:                jz        part05_e
                     ;nein
[000]  323:
[S1E]  324:                mov       dx,vector_segm
                     ;Segment umwandeln
[2C7]  325:                mov       di,offset vector +
           2
[YZ4]  326:                call      conv2ascii
[VED]  327:                mov       dx,vector_offs
                     ;Offset umwandeln
[Y22]  328:                inc       di
[YZ4]  329:                call      conv2ascii
[DKF]  330:                mov       dx,offset vector
                     ;Vektor-Adresse anzeigen
[NE4]  331:                call      display
[VXD]  332:                mov       counter,6
                     ;nur 1 Vektor pro Zeile
[A05]  333: part05_e:      pop       ax
[1SB]  334: part05_f:      inc       al
                     ;Interrupt =+ 1
[WD3]  335:                or        al,al
[HFD]  336:                jz        part05_g
                     ;letzter Int. erreicht ?
[000]  337:
[OLE]  338:                jmp       part05_a
                     ;nächsten Interrupt testen
[000]  339:
[RUF]  340: part05_g:      mov       segm_size,0
                     ;Programm-Grösse = 0
[SMO]  341:                ret
[000]  342:
[HW8]  343: part06:        push      ds
                     ;ES = DS
[T52]  344:                pop       es
[F9K]  345:                mov       ax,memory_size
                     ;bisher gezählte freie Bytes
[PBG]  346:                mov       di,offset prgm_size
           +6      ;letzte Stelle Anzeige
[Y9G]  347:                mov       si,offset prgm_size
                     ;erste Stelle Anzeige
[2RD]  348:                call      conv2dec
                     ;in Dezimalzahl wandeln
[108]  349:                mov       dx,offset underline
[G4C]  350:                call      display
                     ;Striche ausgeben
[W58]  351:                mov       dx,offset free_mem_
           1
```

```
[YKA]  352:                call      display
                    ;Text 'noch '
[028]  353:                mov       dx,offset prgm_size
[96D]  354:                call      display
                    ;Anzahl freie Bytes
[268]  355:                mov       dx,offset free_mem_
       2
[HHD]  356:                jmp short display
                    ;Text 'Bytes frei'
[000]  357:
[HCL]  358: crlf:          mov       dx,offset crlf_char
                    ;Zeilenvorschub ausgeben
[B3D]  359: display:       mov       ah,09h
                    ;DOS-Funktion 9
[Z1C]  360:                int       21h
                    ;String bis '$' ausgeben
[SMO]  361:                ret
[000]  362:
[XXE]  363: conv2ascii:    mov       cx,4
                    ;16-bit Wert aus DX in
[TEF]  364: conv2ascii_a: push       cx
                    ;ASCII wandeln und ab
[6CA]  365:                mov       cl,4
                    ;ES:[DI] speichern
[2R3]  366:                rol       dx,cl
[8P3]  367:                mov       al,dl
[LT9]  368:                and       al,0fh
                    ;4 Bit isolieren
[YU2]  369:                add       al,'0'
[506]  370:                cmp       al,'9'
                    ;0 .. 9
[XR5]  371:                jle       conv2ascii_b
[000]  372:
[DZ5]  373:                add       al,7
                    ;A .. F
[3A8]  374: conv2ascii_b: mov       [di],al
[Y22]  375:                inc       di
[762]  376:                pop       cx
[HDC]  377:                loop      conv2ascii_a
                    ;nächste Ziffer
[000]  378:
[SMO]  379:                ret
[000]  380:
[51F]  381: conv2dec:      mov       dx,ax
                    ;16-bit Wert aus AX (Para-
[AOE]  382:                and       ax,0fffh
                    ;graphen) zu 32-bit Wert in
[BKE]  383:                and       dx,0f000h
                    ;DX:AX (Bytes) machen und in
[DUC]  384:                mov       cl,04h
                    ;Dezimalzahl umwandeln
[CP3]  385:                shl       ax,cl
[899]  386:                shr       dx,cl
                    ;Grösse =* 16
[6X3]  387:                xchg      dh,dl
[6VC]  388:                mov       bx,dx
                    ;Grösse jetzt in BX:AX
[T3A]  389:                std
                    ;absteigend zählen
[FMG]  390: conv2dec_a:    mov       cx,000ah
                    ;in Dezimalzahl wandeln
[MU3]  391:                xor       dx,dx
[514]  392:                xchg      ax,bx
[9RB]  393:                div       cx
                    ;durch 10 dividieren
[514]  394:                xchg      ax,bx
[9RB]  395:                div       cx
                    ;durch 10 dividieren
[H14]  396:                xchg      ax,dx
[WF9]  397:                or        al,'0'
                    ;in ASCII wandeln
[TV9]  398:                stosb
                    ;und speichern
[CU3]  399:                mov       ax,dx
[9K3]  400:                or        bx,bx
[XAC]  401:                jnz       conv2dec_a
                    ;noch nicht fertig
[000]  402:
[KZC]  403:                or        ax,ax
                    ;immer noch nicht fertig
[DP4]  404:                jnz       conv2dec_a
[000]  405:
[HKE]  406:                mov       al,' '
                    ;mit Leerstellen nach links
[ZRB]  407: conv2dec_b:    stosb
                    ;auffüllen
[YVA]  408:                cmp       di,si
                    ;bis Feldanfang
[9D4]  409:                ja        conv2dec_b
[000]  410:
[SMO]  411:                ret
[000]  412:
[E73]  413: code           ends
[000]  414:
[414]  415:                end       tsrmap
```

**Listing. Mit diesem Assemblerlisting haben Sie Überblick über Ihren Arbeitsspeicher**


| | |
|---|---|
| Copy II Option Board *DELUXE* V5.11 | DM 299,– |
| PC Tools V 4.21 | DM 129,– |
| Copy II PC V5 | DM 79,– |
| Utah Cobol V4 | DM 99,– |
| Utah Fortran | DM 99,– |
| Turbo Pascal 4 | DM 169,– |
| Psion Chess dt. | DM 59,95 |
| Flight Sim. II | DM 89,– |
| Balance of P. | DM 64,95 |
| JPI-Modula | DM 199,– |
| Aztec C V4.1 P. | DM 299,– |
| Turbo C V1.5 | DM 169,– |

Kostenlose Prospekte auch für Atari ST und Amiga von

CWTG

Computerversand CWTG
Joachim Tiede Bergstr. 13
D-7109 Roigheim
Tel./BTX 0 62 98/30 98
von 17 - 19 Uhr

SCHRIFTLICHE HÄNDLERANFRAGEN ERWÜNSCHT

**Tools f. Clipper/dBase**

**FormGen*** 398.–
dyn. Formular-Gen. f. dBase/ Clipper, deutsch, 3 Disk.

**HelpEdit*** 198.–
Helpmodul f. Clipper, schreiben Sie Hilfe im fertigen .EXE-Progr.

**Clipper S87*** 1.398,–
Nantucket m. deut. Anpassung

**Netzwerk D-Link*** 1.650.–
einsatzf. f. Clipper/ dBase u. a., incl. Software, kompat. zu Novell o. Token, für 2 Rechner kpl.

**Demos erhältlich DM 15.–**

| | |
|---|---|
| Turbo-C 1.5* | 239.– |
| Turbo-C-Runtime* | 429.– |
| Turbo-Pascal 4.0* | 239.– |
| Turbo Database4.0* | 189.– |
| Turbo-Editor 4.0* | 189.– |
| Turbo-Power 4.0* | 239.– |
| Turbo-Prolog | 239.– |

Alle Turbo-Produkte in Deutsch und Engl. lieferbar

Fachbücher zu dBase / Clipper u. Turbo-Produkten Liste anfordern

Alle mit * gekennzeichneten Produkte werden von uns eingesetzt. Daher leisten wir entsprechenden Support, helfen bei Problemen weiter, natürlich auch telefonisch.

Erstellung individueller Software n. Ihren Wünschen in C, dBase, Clipper

Firma **Elke Zimmermann**
Schütt 12, 6730 Neustadt/Wstr.
☎ 06321/31748

K&P
GESELLSCHAFT FÜR DATENVERARBEITUNG MBH
MÜHLENBERGSTR. 1
6607 QUIERSCHIED
TELEFON: 0 68 97/6 75 00-1
BÜRO AACHEN
TELEFON: 02 41/2 21 66

**dBÜ – dBASE NACH C-ÜBERSETZER**
379,– DM

**dBÜ-JUNIOR – dBASE NACH C-ÜBERSETZER**
269,– DM

**dbFAST – MIT ZUS. DT. ARBEITSBUCH**
249,– DM

**POWER C**
149,– DM

**POWER C (INKL. LIBRARY SOURCE)**
199,– DM

**CLIPPER S'87**
1.099,– DM

**CLIPPER S'87 (d)**
1.799,– DM

**FOXBASE PLUS 2.1 (NEUE VERSION)**
599,– DM

**FOXBASE PLUS 2.1 MULTIUSER**
745,– DM

**FOXBASE PLUS UNIX/XENIX**
1.649,– DM

**R & R CLIPPER/FOXBASE MODULE**
148,– DM

**R & R RELATIONAL REPORTWRITER**
398,– DM

**MS C 5.1**
798,– DM

**MS MACROASSEMBLER 5.1**
278,– DM

**MS QUICK C**
174,– DM

**MS QUICK BASIC**
174,– DM

**MS WORD 4.0 (d)**
1.092,– DM

**SAYWHAT?!**
118,– DM

**TURBO PASCAL 4.0**
164,– DM

**TURBO PASCAL 4.0 (d)**
277,– DM

**TURBO C 1.5**
164,– DM

**TURBO C 1.5 (d)**
277,– DM

**ALLE BORLAND TURBO TOOLS JE STÜCK**
159,– DM

**Auszug aus unseren Geschäftsbedingungen:**
Durch Öffnen der versiegelten Diskettenverpackung werden die jeweils beiliegenden Lizenzbedingungen des Herstellers anerkannt. Rücksendung oder Umtausch von Produkten, die nicht original verpackt sind, ist nicht möglich. Die Gewährleistung beschränkt sich (nach unserer Wahl) auf Ersatzlieferung oder Nachbesserung. Insbesondere ist die Haftung für Mangelfolgeschäden ausgeschlossen.

# Ein Weg zum gekonnten Layout

**Desktop Publishing ist in aller Munde – Anwender, Journalisten und Hersteller schwärmen von den Vorteilen dieser Software. Sollte dieses Thema für Sie noch ein Buch mit sieben Siegeln sein, wird Ihnen der folgende Artikel helfen, die Geheimnise des DTP zu lüften.**

Textverarbeitung ist ohne jede Frage die verbreitetste Anwendung am Computer. Jeder, der schon einmal einen Bericht oder ein anderes wichtiges Dokument mit einer Schreibmaschine zu Papier bringen mußte, weiß die Vorzüge eines Textverarbeitungssystems zu schätzen. Sicherlich ist eine Textverarbeitung ein klarer Fortschritt gegenüber einer Schreibmaschine. Die logische Weiterentwicklung der Textverarbeitung am Computer ist danach Desktop Publishing!

Trotz aller Korrekturmöglichkeiten mit einem Textsystem ist es oftmals schwer bis geradezu unmöglich, das Aussehen der fertigen Seite oder des gesamten Dokuments vorher zu bestimmen. Da hilft meistens nur eines – ein Ausdruck! Anstatt mit dem Computer Papier zu sparen, verbraucht man immer mehr. Mit Desktop-Publishing-Software erhalten Sie vor dem Ausdruck einen ersten Eindruck vom Aussehen des Endprodukts auf dem Papier. Dieses Prinzip wurde von den Herstellern unter dem Kürzel WYSIWYG zusammengefaßt und bedeutet »What you see is what you get« (Was Sie sehen, ist das, was Sie bekommen). Mit Desktop Publishing hielten Bilder und Grafiken Einzug in die Dokumente der Benutzer, denn jedes DTP-Programm kann Grafiken in den Text einbinden. Zu dieser Fähigkeit kamen unter anderem zahlreiche Gestaltungsmöglichkeiten durch Raster, Linien, Spalten und Schriften.

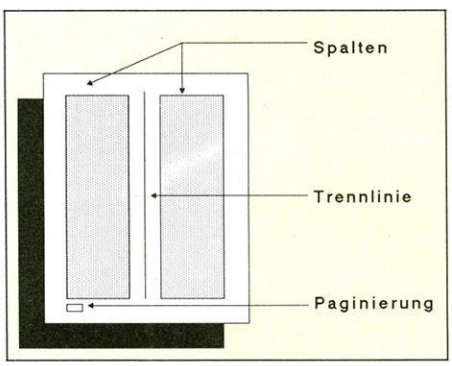



**Bild 1. Die typischen Elemente einer Seite**

## Desktop Publishing ist ein mächtiges Werkzeug – lernen Sie es zu beherrschen

Es ist anzumerken, daß der Umgang mit Desktop Publishing gelernt sein will, denn diese Applikation ist ein mächtiges Werkzeug. Für den Einsteiger gibt es einige einfache Richtlinien, die er beachten sollte, damit er »mehr« aus seiner Seite holen kann.

Das wichtigste beim Anfertigen einer Dokumentation oder einer einzelnen Seite ist die Planung. Machen Sie sich vor der Arbeit an Ihren Texten und Grafiken eine Skizze der einzelnen Seiten (wo kommt was hin) und einen Plan der benötigten Elemente. Das erleichtert es Ihnen sicherlich, Ihren Terminplan bei wichtigen Projekten auch einzuhalten. Mit der Liste können Sie jederzeit feststellen, was Sie noch brauchen, beziehungsweise wie weit Sie Ihre Publikation bereits fertiggestellt haben. Auf den Skizzen der einzelnen Seiten sollten Sie jetzt schon die verschiedenen Elemente verteilen. Legen Sie außerdem auf einem Extrabogen, entsprechend dem Originalformat Ihrer Publikation, die Bemaßung Ihrer Seiten fest. Zu diesen Maßen gehört Papierrand, Spaltenbreiten, -anzahl, Paginierung (Seitenzahl), Position von Linien oder Kästen. Bild 1 zeigt Ihnen die grundsätzlichen Elemente einer Seite.

Notieren Sie sich auf einem weiteren Blatt, welche Schriftarten Sie für einzelne Texte verwenden wollen. Das ist eine ganze Menge Arbeit für den Anfang! Doch Sie werden feststellen, es lohnt sich »vorauszudenken«.

Da gerade von der Planung die Rede ist, benutzen Sie für die Gestaltung Ihrer Seiten nicht zuviele Schriften – dies ist nämlich ein häufiger Fehler von DTP-Anwendern! Zuviele Schriften lenken das Auge nur ab, und so wirkt Ihre Seite »unruhig«. Sollten Sie den Wunsch hegen, mehrere Schriften zu benutzen, dann wählen Sie nur Fonts aus einer Familie. Bekannte Schriftenfamilien sind Helvetica oder Times. Innerhalb einer »Familie« gibt es verschiedene Mitglieder, die unterschiedliche Attribute aufweisen, zum Beispiel fett oder kursiv.

## Ohne Planung ist vernünftige Seitengestaltung nicht möglich

Wenn Sie sich einmal einzelne Schriften genau ansehen, werden Sie feststellen, daß jede einen eigenen »Charakter« besitzt. Genau nach diesem Charakter richtet sich ihr Anwendungsgebiet. Wer würde schon bei einer Einladung zu einem seriösen Geschäftsessen eine verspielte New-Wave-Schrift wählen oder seine Geburtstagskarte mit Trauerbalken und klotzigen Lettern verzieren?

## Bleiben Sie bei einer »Familie«, wenn Sie mit Schrifttypen arbeiten!

Wenn Sie auf einer Seite mit Überschriften arbeiten, achten Sie darauf, daß diese sich nicht gegenseitig die Aufmerksamkeit des Lesers streitig machen. Setzen Sie lieber eine »Hauptüberschrift« und kleinere Zwischenüberschriften ein. Verwenden Sie für Überschriften einen sachlichen und prägnanten Schrifttyp, so »springen« sie dem Leser ins Auge. Für den »normalen« Text (unter Fachleuten Fließtext genannt) benutzen Sie am besten eine gut erfaßbare Schrift. Dazu eignet sich besonders gut eine Schrift mit Serifen, wie Sie in Bild 2 zu sehen ist. Serifen sind übrigens die kleinen »Häkchen«. Zum Vergleich ist auch eine Schrift ohne Serifen in Bild 2 zu sehen. Sollten Sie eine kleine Zeitschrift oder

**Serifen machen eine Schrift lesbarer**

Mitteilung produzieren, fügen Sie nach der Überschrift und dem eigentlichen Text einen sogenannten Vorspann ein. Dieser dient als kurze Einleitung zu dem Text und macht Ihr Dokument »professioneller«.
Im DTP-Bereich werden alle Schriften in »Punkt« gemessen. Diese Einheit ist wesentlich kleiner als Zentimeter und Millimeter – und deshalb für feine Abstufungen viel besser geeignet. Eine Schrift von 9 bis 10 Punkt ist für Zeitschriften üblich, 12 bis 14 Punkt eignet sich für Zwischenüberschriften und 20 bis 36 Punkt bietet sich für Überschriften an.

## Ihre Gestaltung entscheidet über die Qualität Ihres Dokuments

Die Verwendung von Grafiken und gescannten Bildern kann das Aussehen Ihrer Publikation verbessern – oder auch verschlechtern! Das kommt ganz darauf an, wie Sie die einzelnen Elemente zusammenstellen. Das Gefühl für eine gute Gestaltung können Sie sich nicht anlesen, sondern es erfordert viel Übung und Routine im Umgang mit Ihren Werkzeugen. Betrachten Sie also Ihre Seiten kritisch!

```
Jedes Textverar-
beitungsprogramm kann
den Test linksbündig
formatieren.

        Dieser Abschnitt ist
                rechtsbündig
                 formatiert.

Beide Varianten werden
auch als Flattersatz
bezeichnet. Übrigens,
dieser Absatz wurde als
Blocksatz definiert.
```

**Bild 3. Die verschiedenen Satzformatierungen**

Überprüfen Sie, ob Sie die Grafiken nicht zu sehr zwischen Textspalten »eingequetscht« haben. Natürlich ist auch das Gegenteil möglich, die Grafik erdrückt die für Sie wichtige Überschrift. Sollten Sie Bilder mit Gesichtern in Großaufnahme verwenden, achten Sie bitte darauf, daß die Blickrichtung der abgebildeten Personen nicht aus der Publikation herausführt, sondern auf die Heftmitte gerichtet ist.

Damit Sie sich die notwendige Übung verschaffen, gehen Sie einfach folgerndermaßen vor: Sie wählen sich einige typische Elemente einer Seite aus (einen Text, Überschriften und Grafiken) und gestalten mehrere Übungsseiten. Setzen Sie verschiedene Akzente; so soll zum Beispiel das Bild als Aufmacher dienen und der Text nur wenig Platz einnehmen.

## Mit Sparsamkeit an Schriften erreicht man oft mehr

Versuchen Sie sich auch einmal mit Linien. Mit solchen Linien können Sie beispielsweise Spalten trennen. Besonders wenn Sie Ihren Fließtext als links- oder rechtsbündigen Flattersatz definiert haben (Bild 3).

Flattersatz bedeutet nichts anderes, als daß der Text zu einer Seite nicht bündig ist. Das Gegenteil dazu ist natürlich der Blocksatz, bei dem beide Ränder der Spalte bündig sind. Doch zurück zu den Linien: Passen sie deren Breite den zu verwendenden Buchstaben an. Bild 4 zeigt Ihnen deutlich, wie sehr eine zu »starke« Linie stören kann, beziehungsweise daß Sie unbedingt zur Schrift passen muß.

Sie sehen, daß eine ganze Menge dazugehört, ein wirksames Seitenlayout

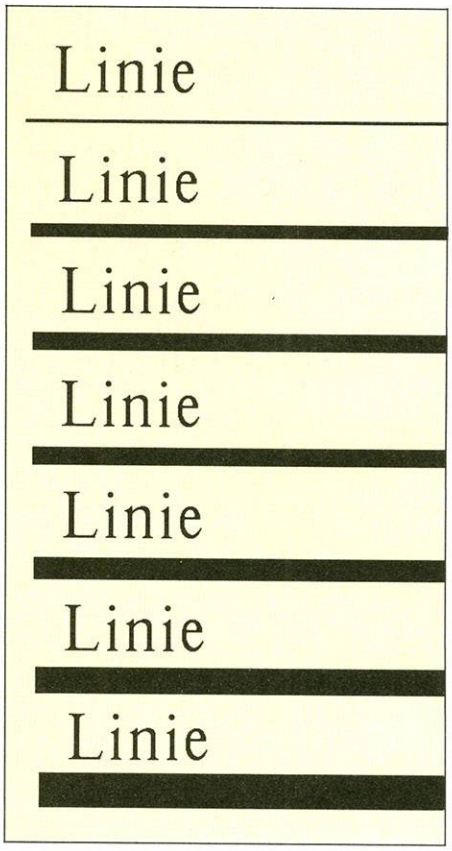



**Bild 4. Linie und Schrift müssen zusammenpassen**

anzufertigen. Als Werkzeug dazu dient Ihr Desktop-Publishing-Programm – und Ihr ästhetisches Empfinden.

Am besten, Sie probieren herum und überprüfen dann kritisch das Ergebnis. Das ist keine verlorene Zeit. Sie lernen das Programm besser kennen und können die Wirkung der verschiedenen Schriften analysieren.

(dm)

**Wer — wo — wann?**

Bei Zuschriften bitten wir Folgendes zu beachten: Haben Sie Fragen, Anregungen oder Programme, die Sie der DOS-Redaktion zukommen lassen möchten, adressieren Sie wie folgt:

**Redaktion · DOS International · Hans-Stießberger-Str. 3 · 8013 Haar/München · Tel. (089) 46 10 01-0**

Haben Sie Fragen, die zum Beispiel Abonnements, Kleinanzeigen oder Databox-Bestellungen betreffen, wenden Sie sich bitte an den Verlag:

**DMV Verlag GmbH · DOS International · Fuldaer Straße 6 · 3440 Eschwege**
**Tel. (0 56 51) 80 09-0 · Telex 9 93 210 dmv d · Telefax (05 65 1) 80 09-33**

# Desktop Publishing zum Abtippen

**Mit SDTP besitzen Sie ein abgerundetes Programmpaket, das es Ihnen erlaubt, Schriftarten zu erzeugen und diese zu verändern, Schnappschüsse des Bildschirminhalts herzustellen und alle so erarbeiteten Grafik- und Schriftdateien in das Hauptprogramm »SDTP« zu integrieren.**

SDTP ist ein Desktop Publisher, mit dem Sie ein Layout für DIN-A4-Seiten bearbeiten können. Mit SDTP können Sie auch Texte und Grafiken nach Ihren eigenen Vorstellungen kombinieren. Sie müssen lediglich Texte als ASCII-Dateien mit der Dateikennung »txt« versehen. Grafiken, die in eine Seite eingebunden werden sollen, sind unter GEM als Img-Datei von Gem Paint oder mit der Snapshot-Funktion in einer beliebigen GEM-Applikation zu speichern. Bei dem abgedruckten Programm »Snapshot« entstehen selbstverständlich auch keine Probleme. Sie können also Ihre Text- und Grafikprogramme wie bisher weiterbenutzen. Die verschiedenen Schrifttypen von GEM (auch die Druckerschriften) verarbeitet SDTP (Listing 1) ebenfalls. Sie können sie allerdings auch mit dem Programm »Fonted« (Listing 2) modifizieren. Oder Sie erstellen mit »Sysfont« (Listing 3) eigene Schriftarten.
SDTP benötigt diese Schrifttypen, daher müssen sie bei der Installation mit auf die Programmdiskette von SDTP kopiert werden. Mit SDTP laden Sie dann diese Texte und Grafiken und positionieren sie auf der dargestellten Bildschirmseite. Die verschiedenen GEM-Schriften und die von Ihnen selbst aus »Sysfont« definierten stehen für Ihre Texte zur Verfügung, die sich auch kursiv und unterstrichen ausgeben lassen. Des weiteren verfügen Sie über die Auswahl zwischen Blocksatz, zentrierter und linksbündiger Darstellung. Überschriften werden auch über die Tastatur eingegeben und positioniert. Sie können selbstverständlich auch Textpassagen und Grafiken durch Linien trennen, Texte und Grafiken verschieben, kopieren und löschen. Ist die Seite komplett zusammengestellt, so können Sie sie auf Diskette ablegen und ausdrucken. In dieser Form abgelegte Seiten lassen sich später wieder laden und weiterbearbeiten. Da Turbo Pascal nur Strings in der Länge von 255 Buchstaben verarbeiten kann, sollten Sie spätestens hier ein Return eingeben.
Das Programm arbeitet nach dem WYSIWIG-Prinzip, das heißt, die Darstellung der Seite auf dem Bildschirm entspricht genau dem Hardcopyausdruck.
Für die Arbeit mit SDTP sind folgende Hardwarevoraussetzungen nötig:
- ein IBM-PC, -XT, -AT oder ein kompatibler Computer
- eine CGA-Grafikkarte
- eine Microsoft-Maus oder eine dazu kompatible
- ein Epson-Drucker oder ein dazu kompatibler

Folgende Softwarevoraussetzungen sind zu erfüllen:
- Turbo Pascal zum Compilieren des Programms
- Das Betriebssystem MS-DOS und das Programm »graftable.exe«, das Sie vor Programmstart aufrufen müssen
- optional die GEM-Schriftdateien
- optional ein Zeichenprogramm unter GEM (beispielweise Gem Paint), das Dateien mit der Endung »img« ablegen kann, oder GEM Snapshot
- Ein Textverarbeitungsprogramm (Edlin von MS-DOS ist ausreichend)

Vergessen Sie nicht, vor dem Programmaufruf das MS-DOS Dienstprogramm »graftabl.exe« aufzurufen, damit SDTP die deutschen Umlaute darstellt.

Das Hauptmenü rechts oben bietet Ihnen folgende Befehle:
- »disk«
- »draw«
- »block«
- »plot«
- »setup«
- »quit«

Wenn Sie einen dieser Befehle mit Ihrer Maus anklicken, öffnet sich ein Pull-Down-Menü, das Ihnen ein Submenü vorschlägt. Bei erneutem Anklicken dieser Befehle blendet sich das Pull-Down-Menü wieder aus. Ist ein Untermenübefehl angewählt, so erscheint dieser links unten auf dem Bildschirm und ist somit aktiv.
Grundsätzlich wird jeder Befehl durch Betätigen der rechten Maustaste verlassen. Danach steht wieder ein neuer Befehl aus dem Hauptmenü rechts oben zur Wahl.
Bei Lesezugriffen auf Disketten zeigt SDTP zuerst die Dateinamen an, die Sie in den Speicher einlesen können. Wählen Sie dann eine Datei mit der Maus, indem Sie den Namen anklicken. Es sind nur zehn Dateien dargestellt. Wenn Sie jedoch einen der beiden Pfeile (oben und unten rechts vom Kasten) anklicken, zeigt Ihnen SDTP weitere Dateinamen an.
Vor Unterkatalogen und Laufwerksbuchstaben steht ein Karo. Klicken Sie dieses an, so wird der Katalog beziehungsweise das Laufwerk gewechselt. Soll der Ladevorgang abgebrochen werden, ist ebenfalls nur der entsprechende Punkt anzuklicken.
- Die Fragekästen in der Mitte des Bildschirms beantworten Sie wie bei GEM mit der Maus, das heißt, es wird eines der drei Kästchen, die Sie rechts im jeweiligen Fragekasten sehen, angeklickt.
- Wird der Mauszeiger sichtbar, so erwartet das Programm eine Mauseingabe.
- Steht nach dem Befehl in der vorletzten Zeile ein Doppelpunkt, so wartet das Programm auf eine Eingabe über die Tastatur.
- Ein anderer Befehl läßt sich erst dann eingeben, wenn in der vorletzten Zeile kein Befehl steht.

Was bewirken nun die Befehle im einzelnen? Zunächst zum Hauptmenüpunkt »disk«:
- »load«: Wenn Sie die aktuelle Seite noch nicht gesichert haben, so macht sie das Programm darauf aufmerksam. Ist dies jedoch nicht der Fall, können Sie eine bereits mit SDTP abgespeicherte Seite laden. Dabei wird die aktuelle Seite gelöscht.
- »save«: Hier fordert Sie das Programm zur Eingabe eines Dateinamens über die Tastatur auf. Danach erfolgt die Speicherung der aktuellen Seite. Existiert bereits eine Seite mit dem angegebenen Namen, so ist das Überschreiben dieser Datei zu bestätigen.
- »font«: Sie müssen eine Fnt-Datei auswählen. SDTP lädt die entsprechende GEM-Schriftdatei oder eine von Ihnen angelegte Sysfont-Datei, diese läßt sich bei jedem neuen Laden von Texten beziehungsweise beim Setzen von Überschriften verwenden.
- »ldtxt«: Mit diesem Befehl laden Sie eine schon bestehende Textdatei. Den Inhalt können Sie wieder auf der Seite positionieren. Danach definieren Sie ein Feld auf der Seite, in das Sie den Text schreiben wollen. Hierzu wählen Sie zuerst einen Eckpunkt auf der Seite aus, indem Sie die linke Maustaste betätigen. Das auf diese Weise erzeugte Rechteck läßt sich mit der Maus bezüglich der Größe abändern. Ist es groß genug, so drücken Sie nochmals die linke Maustaste. Nach Beantwortung einiger Fragen über Textdarstellung (also Blocksatz, zentriert oder linksbündig; ein-, zwei- oder dreispaltig), lädt SDTP soviel Text, wie in dem gewählten Feld Platz haben. Danach wählen Sie erneut ein Feld und laden den Rest des Textes. Mit der rechten Maustaste unterbrechen Sie den Befehl »ldtxt«.

– »ldimg«: Zunächst wählen Sie eine Img-Datei aus. Daraufhin erscheint ein Rechteck in der Größe der zu ladenden Grafik, das sich über die Maus bewegen läßt. Befindet sich das Rechteck an der gewünschten Position, so betätigen Sie die linke Maustaste. SDTP lädt die Grafik und stellt sie an der gewählten Position dar.

Der Hauptmenüpunkt »draw« verbirgt folgende Befehle:
– »u-line«: Aktiviert beziehungsweise deaktiviert die Unterstreichfunktion bei den Befehlen »wrtxt« und »ldtxt«.

– »kursiv«: Schaltet bei den Befehlen »wrtxt« und »ldtxt« die kursive Schrift ein oder aus.

– »trans«: Jeder der folgenden Befehle wird im Transparent-Modus ausgeführt, das bedeutet, es wird die alte Grafik oder der alte Text von der neuen Grafik beziehungsweise dem neuen Text überlagert.

– »wrtxt«: Sie können eine Überschrift über die Tastatur eingeben. Danach ist sie mit der Maus auf der Seite zu positionieren und durch Drücken der linken Maustaste darzustellen.

– »line«: Sie bestimmen einen Anfangspunkt auf der Seite durch das Betätigen der linken Maustaste. Den Endpunkt der Linie setzen Sie durch nochmaliges Drücken der linken Maustaste fest. Es werden so lange horizontale oder vertikale Linien gezogen, bis Sie die rechte Maustaste betätigen.

Die Befehle des Hauptmenüpunkts »block«:
– »set«: Sie definieren einen Block genauso, wie Sie beispielsweise bei dem Befehl »ldtxt« ein Feld auswählen. Sie können Blöcke nur bewegen, löschen oder kopieren. Bei den Befehlen »ldtxt«, »ldimg« und »wrtxt« wird automatisch einen Block in der Größe des Textes beziehungsweise der Grafik festgelegt.

– »del«: Ein bestehender Block und sein Inhalt sind dadurch zu löschen, daß Sie die Maus in das entsprechende Rechteck bewegen und dann die linke Maustaste einsetzen.

– »copy«: Einen bestehenden Block kopieren Sie, indem Sie ihn wie beim Befehl »del« anwählen und dann das angezeigte Rechteck – das mit der Maus zu bewegen ist – mit der linken Maustaste an einer anderen Stelle plazieren.

– »move«: Sie verschieben einen Block in derselben Weise, in der Sie ihn kopiert (»copy«) haben. Der Unterschied zwischen den Befehle besteht darin, daß Sie bei »move« das Original löschen.

– »delall«: Nach Anwählen dieses Befehls löschen Sie die gesamte Seite. Wenn Sie die Seite noch nicht gespeichert haben, macht Sie SDTP darauf aufmerksam, und Sie entscheiden, ob Sie das erstellte Layout wirklich löschen wollen oder nicht.

– »plot«: Sie geben die Seite auf dem Drucker aus.
Die Einstellungsmöglichkeiten von »setup«:

– »save«: Sie speichern die für den Programmstart eingegebene Schriftwahl.

– »load«: Sie laden die mit dem Befehl »save« abgelegte Schriftwahl.
– »font«: Hier legen Sie fest, welchen Schrifttyp Sie nach dem Programmstart laden wollen. Die gewählte Schriftdatei muß sich auf der SDTP-Programmdiskette befinden. Diese Einstellung ist mit dem Befehl »save« zu sichern, bevor Sie die Schrift automatisch beim Start von SDTP laden können.

– »drive«: Sie müssen dem Programm mitteilen, welche zusätzlichen Laufwerke neben den Diskettenlaufwerken A und B vorhanden sind. Laufwerk C klicken Sie an, wenn Sie entweder eine RAM-Disk eingerichtet haben oder über eine Harddisk verfügen, C und D entsprechend, wenn Sie beides oder aber zwei Harddisks besitzen.

– »quit«: Mit diesem Befehl verlassen Sie SDTP. Das Programm macht Sie gegebenenfalls darauf aufmerksam, daß Sie die aktuelle Seite noch nicht abgespeichert haben.

Wie gehen Sie nun vor, um eine SDTP-Programmdiskette anzulegen? Zunächst compilieren Sie die Datei »sdtp.pas« mit Turbo Pascal 3.xx und speichern sie als Com-Datei ab. Sie legen die Diskette mit »sdtp.com« ein und gehen folgendermaßen vor:

```
copy con SDTP.set [Return]
A>copy con SDTP.set
X [Ctrl-Z] [Return]
X↑Z
1 Datei(en) kopiert
```

Nun legen Sie die MS-DOS-Diskette ein:

```
Graftabl [Return]
A>Graftabl
Graphik Zeichen geladen
```

Falls Sie die Schriftdateien von GEM benutzen wollen, müssen Sie die folgenden Zeilen eingeben, sonst kopieren Sie Ihre mit »Sysfont« erstellten Schriftdateien auf die Diskette.

```
xcopy gemsys\*.fnt /w /p [Return]
A>xcopy gemsys\*.fnt /w /p
Zum Kopieren beliebige Taste drücken
```

Jetzt legen Sie die GEM-Diskette mit Fnt-Dateien ein und drücken irgendeine Taste:

```
gemsys\xxxxxxxx.fnt (J/N)?
```

Diskette mit »sdtp.com« einlegen und jeden Dateinamen mit »j« bestätigen

```
x Datei(en) kopiert
SDTP [Return]
A>SDTP
```

## SDTP ist an Gem orientiert

Durch dieses Vorgehen laden Sie SDTP und erhalten die Meldung angezeigt, daß die entsprechende Fnt-Datei nicht gefunden wird. Geben Sie mit der linken Maustaste »OK« an. Daraufhin klicken Sie den Menüpunkt »setup« an und wählen im Pulldown-Menü den Befehl »font« aus. Suchen Sie nun eine Schriftdatei aus (siehe Befehlsbeschreibung »Font« unter »Setup«). Klicken Sie dann nochmals »setup« und im Pulldown-Menü den Punkt »drive« an. Geben Sie nun ihre zusätzlichen Laufwerke bekannt (siehe Programmbeschreibung zu »drive«). Daraufhin sind nur noch diese Daten abzuspeichern. Dazu wird erneut »setup« angeklickt und im Pulldown-Menü »save«. Jetzt haben Sie die Installation abgeschlossen.
Ein weiterer Bestandteil des SDTP-Pakets ist der Fonteditor »Fonted« und das Programm »Sysfont« sowie »Snapshot« für diejenigen Leser, die nicht über GEM verfügen.
Das Programm »Fonted« dient zur Veränderung bereits bestehender GEM-Schriftdateien oder der mit »Sysfont« hergestellten Schriftdatei »sfont.fnt«. Nachdem Sie »Fonted« mit Turbo Pascal compiliert haben, rufen Sie das Programm entweder mit »fonted« oder »fonted [Schriftname]« auf. Verfahren Sie nach der ersten Methode, so fordert Sie das Programm auf, eine Schriftdatei anzugeben. Diese können Sie auch ohne die Kennung »fnt« eingeben. Im zweiten Fall lädt »Fonted« die angegebene Schriftdatei sofort und es wird das ASCII-Zeichen 32 (das Leerzeichen) dargestellt. Auf der linken Hälfte des Bildschirms finden Sie die Befehle, die Sie mit der Maus anklicken können. Folgende Auswahl steht Ihnen dabei zur Verfügung:
– »name«, »font«, »points«: Mit diesen Befehlen lassen sich die in der Fnt-Datei gespeicherten Informationen über die Schrift verändern. Die Eingabe erfolgt mit der Tastatur, nachdem Sie den Befehl mit der Maus angeklickt haben.

- »Zeichen löschen«: der aktuelle Buchstabe wird gelöscht
- »Zeichen sichern«: der aktuelle Buchstabe wird gesichert
- »Schrift sichern«: die FNT-Datei wird abgespeichert
- »Ende«: das Programm wird beendet
- »Zeichen«: Sie wählen den aktuelle Buchstaben
- »Ascii«: der ASCII-Wert des aktuellen Buchstabens
- »Zeilen«: die Zeilenzahl wird geändert (für alle Zeichen)
- »Breite«: Sie können die Breite des aktuellen Zeichens ändern

Bei den Befehlen »Zeichen löschen«, »Zeilen« und »Breite« speichern Sie den aktuellen Buchstaben, während bei den Befehlen »Zeichen« und »Ascii« der aktuelle Buchstabe verloren geht. Zuvor müssen Sie »Zeichen sichern« aufrufen, wenn der Buchstabe erhalten bleiben soll. Erst mit dem Befehl »Schrift sichern« legen Sie die Schrift auch wirklich auf Diskette ab (vorher befinden sich alle Ihre Änderungen nur im Arbeitsspeicher).

## Der aktive Befehl ist links unten angegeben

Zwischen Menü und dargestelltem Zeichen auf der rechten Hälfte, in der Sie mit der linken Maustaste Bildpunkte hinzufügen und mit der rechten Maustaste löschen können, befindet sich eine Spalte mit zwei Markierungen. Die obere Spalte gibt Ihnen die Position der Grundlinie an. Die Länge zwischen beiden Markierungen entspricht der Unterlänge. Diese Markierungen können Sie verschieben, indem Sie sie einfach mit der linken Maustaste anklicken. Solange Sie die Taste gedrückt halten, läßt sich die Marke mit der Maus verschieben.

Nachdem Sie den Befehl »Ende« angewählt haben, bestätigen Sie das Beenden des Programms. Hierbei klicken Sie einfach mit der linken Maustaste entweder das »Ja« oder das »Nein« im Kasten an, der in der Mitte des Bildschirms dargestellt ist.

Wenn Sie Grafiken in Ihr Layout einbinden möchten und nicht über GEM verfügen, hilft Ihnen hier das Programm »Snapshot« (Listing 4) weiter.

Dieses Programm ist speicherresident. Sie starten es einmal von der MS-DOS-Betriebssystemebene aus. Sie rufen »Snapshot« mit der Tastenkombination [Shift] und [PrtSc] auf. »Snapshot« dient dazu, Bildschirmausschnitte von anderen Programmen im GEM-Format abzuspeichern, damit Sie diese für SDTP einsetzen können. Das aktuelle Programm, von dem aus Sie »Snapshot« aufrufen, wird unterbrochen, und in der obersten Zeile tippen Sie die Eingabe eines Dateinamens ohne Extension ein. Unter diesem Namen speichert »Snapshot« die Bilddatei dann auf Diskette. Nach der Eingabe des Dateinamens läßt sich mit der Maus ein Kreuz auf dem Bildschirm bewegen, mit dem die linke obere Ecke festgelegt wird. Die Übergabe der Position erfolgt durch die linke Maustaste. Diese halten Sie dann so lange gedrückt, bis das so erzeugte Rechteck, das Sie mit Hilfe der Maus bezüglich der Größe verändern können, dem von Ihnen gewünschten Bildschirmausschnitt entspricht. Sobald Sie die Taste loslassen, wird dieser von Ihnen definierte Ausschnitt auf Diskette abgespeichert.

Das letzte Programm des SDTP-Programmpakets heißt »Sysfont«. Dieses Programm liest den Grafikzeichensatz, der in jedem IBM-kompatiblen PC im BIOS-ROM ab Adresse FFA6Eh vorhanden ist, aus und erzeugt daraus eine Schriftdatei. Da im BIOS-ROM nur die Zeichen 0-127 abgelegt sind, konstruiert das Programm die Umlaute selbst. Die restlichen Zeichen ab 128 sind Leerzeichen. Aufgerufen wird das Programm entweder mit »Sysfont« oder mit »Sysfont [Dateiame]«, wobei Dateiname ein Schriftdateiname ohne Extension sein muß. Unter diesem Namen legt »Sysfont« die neue Schriftdatei an. Geben Sie keinen Namen an, so erhält die erzeugte Schriftdatei automatisch die Bezeichnung »sfont.fnt«. Außerdem ist es möglich, weitere Schrifttypen mit »Fonted« zu generieren, auch wenn Sie nicht über GEM verfügen. Hier ist es empfehlenswert, die Schriftdatei »sfont.fnt« mit dem MS-DOS-Befehl »copy« zu kopieren (beispielsweise mit »copy sfont.fnt eigene.fnt«). Danach rufen Sie den Fonteditor »Fonted« auf (zum Beispiel durch den Befehl »fonted eigene«) und Sie können die Schrift problemlos nach Ihren Wünschen gestalten.

(Ramon Ketzlick/gh)

```
[H51]    0: {$C-}
[2U4]    1: program   SDTP;
[000]    2:
[VM1]    3: const
[KN9]    4:   menu : array[0..5,0..5] of string[6] =
[GFA]    5:          (' DISK ','LOAD  ','SAVE  ','FONT
           ','LdTXT ','LdIMG '),
[09B]    6:          (' DRAW ','U-LINE','KURSIV','TRAN
           S ','WrTXT ','LINE  '),
[9SA]    7:          ('BLOCK ','SET   ','DEL   ','MOVE
           ','COPY  ','DELALL'),
[9T7]    8:          (' PLOT ','      ','      ','
           ','      ','      '),
[WNA]    9:          ('SETUP ','SAVE  ','LOAD  ','FONT
           ','DRIVE ','      '),
[HV7]   10:          (' QUIT ','      ','      ','
           ','      ','      '));
[7S4]   11:   prgname  = '  SDTP';
[1RK]   12:   system : array[0..14] of byte = (14,41,17
          6,182,183,55,16,165,178,58,61,54,180,177,181
          );
[WEF]   13:   leerstr  = '

          ';{72 spaces}
[E26]   14:   setupname= 'SDTP.SET';
[234]   15:   fontpos  = 11;
[VZ3]   16:   ulinepos = 21;
[774]   17:   kursivpos= 28;
[C24]   18:   transpos = 35;
[G94]   19:   maxblock = 255;
[CC4]   20:   maxfont  = 15000;
[2X3]   21:   maxanz   = 128;
[000]   22:
[A21]   23: type
[W15]   24:   regtyp    = record
[S25]   25:                 case integer of
[5FC]   26:                   1:(ax,bx,cx,dx,dp,si,di,d
          s,es,fl : integer);
```

```
[3SA]   27:                   2:(al,ah,bl,bh,cl,ch,dl,d
          h       : byte   );
[TUO]   28:                 end;
[FL5]   29:   inhalttyp = record
[3C2]   30:                 anz  : byte;
[00B]   31:                 name : array[1..maxanz] of
          string[13];
[TUO]   32:               end;
[U15]   33:   dtatyp    = record
[4E9]   34:                 reserviert : array[1..21] o
          f byte;
[R55]   35:                 attribut   : byte;
[0X4]   36:                 zeit       : integer;
[895]   37:                 datum      : integer;
[Z19]   38:                 bytes      : array[1..2] of
          integer;
[4N7]   39:                 name       : array[1..13] o
          f char;
[TUO]   40:               end;
[5E5]   41:   str5      = string[5];
[VW5]   42:   str13     = string[13];
[5W5]   43:   str20     = string[20];
[NW5]   44:   str40     = string[40];
[DY5]   45:   str64     = string[64];
[DX5]   46:   str70     = string[70];
[L16]   47:   str255    = string[255];
[FL5]   48:   seitentyp = record
[W1A]   49:                 daten   : array[1..4,0..79,
          0..85] of byte;
[ZC5]   50:                 blocknr : byte;
[CYB]   51:                 block   : array[0..maxblock
          ] of record
[YG4]   52:
                      xpos   : integer;
[ZG4]   53:
                      ypos   : integer;
[885]   54:
                      breite : integer;
```

```
[VS4]   55:
              hoehe   : integer;
[TUO]   56:
            end;
[TUO]   57:               end;
[DWB]   58:    datentyp  = array[0..maxfont] of byte;
[2P5]   59:    setuptyp  = record
[L93]   60:                  font   : str70;
[NG3]   61:                  drive  : byte;
[TUO]   62:               end;
[000]   63:
[EMO]   64: var
[GM6]   65:    seite       : seitentyp;
[PKD]   66:    video       : array[0..16383] of byte abso
        lute $b800:0000;
[T39]   67:    video1      : array[0..16383] of byte;
[23A]   68:    font        : record daten  : datentyp;
[R95]   69:                          offset : integer;
[985]   70:                          zeilen : integer;
[BW4]   71:                          name   : string[9];
[TUO]   72:                end;
[116]   73:    setup       : setuptyp;
[VV4]   74:    quit        : boolean;
[E55]   75:    uline       : boolean;
[XR5]   76:    kursiv      : boolean;
[775]   77:    trans       : boolean;
[3A6]   78:    seitenflag  : boolean;
[565]   79:    aktmenu     : byte;
[000]   80:
[E59]   81: function mouseinit:boolean;
[KA5]   82: var reg:regtyp;
[BH1]   83: begin
[UG2]   84:    reg.ax:=0;
[SB3]   85:    intr($33,reg);
[BR5]   86:    mouseinit:=false;
[3CB]   87:    if reg.ax=$ffff then mouseinit:=true;
[TUO]   88: end;
[000]   89:
[VZ5]   90: procedure mouseon;
[KA5]   91: var reg:regtyp;
[BH1]   92: begin
[VG2]   93:    reg.ax:=1;
[SB3]   94:    intr($33,reg);
[TUO]   95: end;
[000]   96:
[U76]   97: procedure mouseoff;
[KA5]   98: var reg:regtyp;
[BH1]   99: begin
[WG2]  100:    reg.ax:=2;
[SB3]  101:    intr($33,reg);
[TUO]  102: end;
[000]  103:
[EV8]  104: function xmouse:integer;
[KA5]  105: var reg:regtyp;
[BH1]  106: begin
[XG2]  107:    reg.ax:=3;
[SB3]  108:    intr($33,reg);
[B15]  109:    xmouse:=reg.cx;
[TUO]  110: end;
[000]  111:
[GV8]  112: function ymouse:integer;
[KA5]  113: var reg:regtyp;
[BH1]  114: begin
[XG2]  115:    reg.ax:=3;
[SB3]  116:    intr($33,reg);
[H15]  117:    ymouse:=reg.dx;
[TUO]  118: end;
[000]  119:
[OFA]  120: procedure setmouse(x,y:integer);
[KA5]  121: var reg:regtyp;
[BH1]  122: begin
[YG2]  123:    reg.ax:=4;
[CL2]  124:    reg.cx:=x;
[KL2]  125:    reg.dx:=y;
[SB3]  126:    intr($33,reg);
[TUO]  127: end;
[000]  128:
[9BD]  129: procedure mousewindow(x1,y1,x2,y2:integer);
[KA5]  130: var reg:regtyp;
[BH1]  131: begin
[1H2]  132:    reg.ax:=7;
[9S2]  133:    reg.cx:=x1;
[GS2]  134:    reg.dx:=x2;
[SB3]  135:    intr($33,reg);
[2H2]  136:    reg.ax:=8;
[AS2]  137:    reg.cx:=y1;
[HS2]  138:    reg.dx:=y2;
[SB3]  139:    intr($33,reg);
[TUO]  140: end;
[000]  141:
[3U8]  142: function lbutton:boolean;
[KA5]  143: var reg:regtyp;
[BH1]  144: begin
[XG2]  145:    reg.ax:=3;
[NK5]  146:    lbutton:=false;
[SB3]  147:    intr($33,reg);
[KBC]  148:    if (reg.bx and 1)=1 then lbutton:=true;
[TUO]  149: end;
[000]  150:
```

```
[FU8]  151: function rbutton:boolean;
[KA5]  152: var reg:regtyp;
[BH1]  153: begin
[XG2]  154:    reg.ax:=3;
[VK5]  155:    rbutton:=false;
[SB3]  156:    intr($33,reg);
[1CC]  157:    if (reg.bx and 2)=2 then rbutton:=true;
[TUO]  158: end;
[000]  159:
[207]  160: procedure loadscreen;
[BH1]  161: begin
[E03]  162:    mouseoff;
[LP6]  163:    move(video1,video,16383);
[B03]  164:    mouseon;
[TUO]  165: end;
[000]  166:
[E27]  167: procedure savescreen;
[BH1]  168: begin
[E03]  169:    mouseoff;
[PU6]  170:    move(video,video1,16383);
[B03]  171:    mouseon;
[TUO]  172: end;
[000]  173:
[L3C]  174: procedure xordraw(x1,y1,x2,y2:integer);
[EMO]  175: var
[20C]  176:    xorx,xory,xordx,xordy,xorn1:integer;
[XB4]  177: label xorende;
[P9C]  178: procedure xorplot(yorx,yory:integer);
[EMO]  179: var
[965]  180:    xoradr:integer;
[403]  181:    xorp:byte;
[MT3]  182: label xoende;
[BH1]  183: begin
[B5L]  184:    if (yorx<0) or (yorx>639) or (yory<0) or
       (yory>199) then goto xoende;
[6AD]  185:    if odd(yory) then xoradr:=8152+yory*40+(y
       orx shr 3)
[42B]  186:                 else xoradr:=     yory*40+(y
       orx shr 3);
[OV6]  187:    xorp:=$80 shr (yorx mod 8);
[WLC]  188:    video[xoradr]:=video[xoradr] xor xorp;
[CO3]  189: xoende:end;
[000]  190:
[BH1]  191: begin
[5T4]  192:    xordx:=abs(x1-x2);
[NT4]  193:    xordy:=abs(y1-y2);
[32B]  194:    if xordx+xordy=0 then goto xorende;
[UCA]  195:    if x1<x2 then xorx:=x1 else xorx:=x2;
[FDA]  196:    if y1<y2 then xory:=y1 else xory:=y2;
[B86]  197:    xorplot(xorx,xory);
[DUE]  198:    if xordx=0 then for xorn1:=1 to xordy do
       begin
[7H5]  199:       xory:=succ(xory);
[B86]  200:       xorplot(xorx,xory);
[D8C]  201:    end else for xorn1:=1 to xordx do begin
[WG5]  202:       xorx:=succ(xorx);
[B86]  203:       xorplot(xorx,xory);
[TUO]  204:    end;
[UP3]  205: xorende:end;
[000]  206:
[UCC]  207: procedure fillbox(x1,y1,sx,sy:integer);
[EMO]  208: var
[LS9]  209:    zaehler,zaehler1,adr:integer;
[BH1]  210: begin
[T4B]  211:    for zaehler:=0 to pred(sy shr 1) do begin
[NT9]  212:       adr:=(zaehler shl 1 +y1)*40+(x1 shr 3);
[HM9]  213:       fillchar(video[adr],sx shr 3,$aa);
[WRA]  214:       fillchar(video[adr+$2000],sx shr 3,$55)
       ;
[TUO]  215:    end;
[TUO]  216: end;
[000]  217:
[K99]  218: procedure openmenu(mnr:byte);
[EMO]  219: var
[U45]  220:    mschleife1 : byte;
[W45]  221:    mschleife2 : byte;
[ZU3]  222:    manz       : byte;
[BH1]  223: begin
[432]  224:    manz:=5;
[576]  225:    if mnr=4 then manz:=4;
[E03]  226:    mouseoff;
[038]  227:    for mschleife1:=1 to manz do
[3KE]  228:       for mschleife2:=succ(manz-mschleife1) t
       o manz do
[BH1]  229:          begin
[ECF]  230:             gotoxy(43+mnr*6,mschleife2+mschleife1
       -pred(manz));
[UG8]  231:             write(menu[mnr,mschleife2]);
[OA2]  232:             delay(20);
[TUO]  233:          end;
[B03]  234:    mouseon;
[TUO]  235: end;
[000]  236:
[N2A]  237: procedure closemenu(mnr:byte);
[EMO]  238: var
[U45]  239:    mschleife1 : byte;
[W45]  240:    mschleife2 : byte;
[ZU3]  241:    manz       : byte;
[BH1]  242: begin
```

```
[432]  243:    manz:=5;
[576]  244:    if mnr=4 then manz:=4;
[E03]  245:    mouseoff;
[LLB]  246:    for mschleife1:=pred(manz) downto 1 do
[PLE]  247:      for mschleife2:=succ(manz)-mschleife1 t
       o manz do
[BH1]  248:      begin
[ECF]  249:        gotoxy(43+mnr*6,mschleife2+mschleife1
       -pred(manz));
[UG8]  250:        write(menu[mnr,mschleife2]);
[YKG]  251:        fillbox(340,succ(mschleife2+mschleife
       1-manz) shl 3,240,8);
[0A2]  252:        delay(20);
[TUO]  253:      end;
[C25]  254:    fillbox(340,8,240,8);
[B03]  255:    mouseon;
[TUO]  256:  end;
[000]  257:
[RA9]  258:  procedure copymenu(bnr:byte);
[EMO]  259:  var
[6D4]  260:   cs      : string[6];
[AS3]  261:   xp,yp   : byte;
[4B4]  262:   endpos : byte;
[BH1]  263:  begin
[E03]  264:    mouseoff;
[GA7]  265:    cs:=menu[aktmenu,bnr];
[G64]  266:    xp:=43+aktmenu*6;
[U64]  267:    yp:=succ(bnr);
[BK8]  268:    if ((cs='TRANS ') and trans) or
[LE7]  269:       ((cs='KURSIV') and kursiv) or
[028]  270:       ((cs='U-LINE') and uline) then
[BH1]  271:    begin
[LT8]  272:      if cs='TRANS ' then xp:=transpos;
[S99]  273:      if cs='KURSIV' then xp:=kursivpos;
[1W8]  274:      if cs='U-LINE' then xp:=ulinepos;
[AP1]  275:      yp:=24;
[4S3]  276:      gotoxy(xp,25);
[923]  277:      write('      ');
[564]  278:      gotoxy(xp,yp);
[3N2]  279:      write(cs);
[X52]  280:      repeat
[ZP3]  281:        xp:=succ(xp);
[2M5]  282:        gotoxy(pred(xp),yp);
[KF3]  283:        write(#32,cs);
[853]  284:      until xp=74;
[564]  285:      gotoxy(xp,yp);
[923]  286:      write('      ');
[FP3]  287:      yp:=pred(yp);
[X52]  288:      repeat
[FP3]  289:        yp:=pred(yp);
[8EA]  290:        fillbox(pred(xp) shl 3,yp shl 3,48,8)
       ;
[564]  291:        gotoxy(xp,yp);
[3N2]  292:        write(cs);
[0A2]  293:        delay(20);
[CC5]  294:      until yp=succ(bnr);
[X52]  295:      repeat
[CP3]  296:        xp:=pred(xp);
[UAA]  297:        fillbox((xp+5) shl 3,pred(yp) shl 3,8
       ,8);
[564]  298:        gotoxy(xp,yp);
[3N2]  299:        write(cs);
[AD6]  300:      until xp=43+aktmenu*6;
[Z16]  301:      closemenu(aktmenu);
[LM2]  302:    end else
[BH1]  303:    begin
[X52]  304:      repeat
[ZP3]  305:        xp:=succ(xp);
[LAA]  306:        fillbox((xp-2) shl 3,pred(yp) shl 3,8
       ,8);
[564]  307:        gotoxy(xp,yp);
[3N2]  308:        write(cs);
[853]  309:      until xp=74;
[X52]  310:      repeat
[2R3]  311:        yp:=succ(yp);
[XSA]  312:        fillbox(pred(xp) shl 3,(yp-2) shl 3,4
       8,8);
[564]  313:        gotoxy(xp,yp);
[3N2]  314:        write(cs);
[0A2]  315:        delay(20);
[553]  316:      until yp=24;
[7U2]  317:      endpos:=1;
[TX9]  318:      if cs='TRANS ' then endpos:=transpos;
[YWA]  319:      if cs='KURSIV' then endpos:=kursivpos;
[B2A]  320:      if cs='U-LINE' then endpos:=ulinepos;
[X52]  321:      repeat
[CP3]  322:        xp:=pred(xp);
[564]  323:        gotoxy(xp,yp);
[6G3]  324:        write(cs,#32);
[5K5]  325:      until xp=endpos;
[Z16]  326:      closemenu(aktmenu);
[TUO]  327:    end;
[B03]  328:    mouseon;
[TUO]  329:  end;
[000]  330:
[8Z5]  331:  procedure display;
[EMO]  332:  var
[OR7]  333:    schleife1,schleife2 : byte;
[8R7]  334:    schleife3,schleife4 : byte;
[OM8]  335:    speicherstelle      : integer;
[5U5]  336:    pixel1,pixel2       : byte;
[BH1]  337:  begin
[E03]  338:    mouseoff;
[P76]  339:    for schleife1:=0 to 85 do
[BH1]  340:    begin
[009]  341:      speicherstelle:=schleife1*80;
[U76]  342:      for schleife2:=0 to 39 do
[BH1]  343:      begin
[2E2]  344:        pixel2:=0;
[CE6]  345:        for schleife3:=0 to 1 do
[BH1]  346:        begin
[VCK]  347:          pixel1:=seite.daten[1,schleife2 shl
       1 +schleife3,schleife1] or
[ZOF]  348:                  seite.daten[2,schleife2 shl
       1 +schleife3,schleife1];
[VE6]  349:          for schleife4:=0 to 3 do
[BH1]  350:          begin
[6UG]  351:            pixel2:=((((pixel1 shr 7) or (pix
       el1 shr 6)) and 1) or pixel2);
[6WF]  352:            if (schleife3=0) or (schleife4<3)
       then pixel2:=pixel2 shl 1;
[585]  353:            pixel1:=pixel1 shl 2;
[TUO]  354:          end;
[TUO]  355:        end;
[KFB]  356:        video[speicherstelle]:=pixel2 xor 255
       ;
[2E2]  357:        pixel2:=0;
[CE6]  358:        for schleife3:=0 to 1 do
[BH1]  359:        begin
[5DK]  360:          pixel1:=seite.daten[3,schleife2 shl
       1 +schleife3,schleife1] or
[91F]  361:                  seite.daten[4,schleife2 shl
       1 +schleife3,schleife1];
[VE6]  362:          for schleife4:=0 to 3 do
[BH1]  363:          begin
[6UG]  364:            pixel2:=((((pixel1 shr 7) or (pix
       el1 shr 6)) and 1) or pixel2);
[6WF]  365:            if (schleife3=0) or (schleife4<3)
       then pixel2:=pixel2 shl 1;
[585]  366:            pixel1:=pixel1 shl 2;
[TUO]  367:          end;
[TUO]  368:        end;
[F1D]  369:        video[speicherstelle+$2000]:=pixel2 x
       or 255;
[Z3C]  370:        speicherstelle:=succ(speicherstelle);
[TUO]  371:      end;
[TUO]  372:    end;
[B03]  373:    mouseon;
[TUO]  374:  end;
[000]  375:
[K1D]  376:  procedure setpoint(xpos,ypos:integer);
[EMO]  377:  var
[7P4]  378:    var1,p:integer;
[BH1]  379:  begin
[FNG]  380:    if (xpos>=0) and (xpos<640) and (ypos>=0)
       and (ypos<344) then
[BH1]  381:    begin
[WHA]  382:      if odd(ypos) then var1:=2 else var1:=1;
[8A9]  383:      if (ypos mod 4)>1 then var1:=var1+2;
[PL5]  384:      p:=$80 shr (xpos mod 8);
[F5U]  385:      seite.daten[var1,xpos shr 3,ypos shr 2]
       :=seite.daten[var1,xpos shr 3,ypos shr 2] or
       p;
[EP5]  386:      seitenflag:=true;
[TUO]  387:    end;
[TUO]  388:  end;
[000]  389:
[UKD]  390:  procedure resetpoint(xpos,ypos:integer);
[EMO]  391:  var
[7P4]  392:    var1,p:integer;
[BH1]  393:  begin
[U5N]  394:    if (xpos>=0) and (xpos<640) and (ypos>=0)
       and (ypos<344) and not(trans) then
[BH1]  395:    begin
[WHA]  396:      if odd(ypos) then var1:=2 else var1:=1;
[8A9]  397:      if (ypos mod 4)>1 then var1:=var1+2;
[KY7]  398:      p:=($80 shr (xpos mod 8)) xor 255;
[2RU]  399:      seite.daten[var1,xpos shr 3,ypos shr 2]
       :=seite.daten[var1,xpos shr 3,ypos shr 2] an
       d p;
[EP5]  400:      seitenflag:=true;
[TUO]  401:    end;
[TUO]  402:  end;
[000]  403:
[B9E]  404:  function test(xpos,ypos:integer):boolean;
[EMO]  405:  var
[7P4]  406:    var1,p:integer;
[BH1]  407:  begin
[VGG]  408:    if (xpos>-1) and (xpos<640) and (ypos>-1)
       and (ypos<343) then
[BH1]  409:    begin
[WHA]  410:      if odd(ypos) then var1:=2 else var1:=1;
[8A9]  411:      if (ypos mod 4)>1 then var1:=var1+2;
[PL5]  412:      p:=$80 shr (xpos mod 8);
[4VC]  413:      if seite.daten[var1,xpos shr 3,ypos shr
       2]
```

# FÜR IBM-PCs UND KOMPATIBLE

## Ausgefeilte Programmiertechnik und praxiserprobte Musterprogramme in Assembler

inklusive Diskette

Das Intel 16/32 Bit Assemblerhandbuch bietet Ihnen

■ **systematische Programmierkurse** mit sofort nachvollziehbaren Übungsbeispielen, z.B. zur Stringverarbeitung, Interruptprogrammierung oder Makrotechnik, Assemblerkurs 5.0 und Code-View;

■ **detaillierte Dokumentationen von Architektur und Registern** der Prozessoren 8088 und 80(x)86 sowie der Coprozessoren 80(2)87;

■ **den Befehlsvorrat** dieser Prozessoren: übersichtlich aufbereitet und mit ausführlichen Kommentaren;

■ **Makrobibliotheken:** u.a. Stringbibliothek;

■ **Module eines komfortablen Makroassemblers:** Von Anfang an sind Sie bei der Entwicklung eines Makroassemblers für den 8086-Prozessor dabei. Sie erkennen Aufbau und Funktion der einzelnen Module sowie deren Zusammenwirken. So erhalten Sie Schritt für Schritt ein exzellentes Programmierwerkzeug;

■ **bewährte Musterlösungen in Assembler:**
Sie erhalten **Systemprogramme,** u.a. Fenster für DOS, **Druckerutilities** wie ein Druckerfestprogramm, **Schnittstellenprogramme,** u.a. ein Programm zum Datentransfer zwischen PC-/MS-DOS-Rechnern, **technische Programme** wie das Muster einer Interruptroutine oder die Programmierung des Timers 8253-5, **mathematische Programme, u.a. CRC-Prüfsummen/-wörter für Dateien, Anwenderprogramme** wie Passwörter für Systemdisketten oder Ver- und Entschlüsseln von EXE-Dateien sowie **Utilities** wie zum Dateien-Dump.

■ **Ergänzungsausgaben zum Grundwerk** mit aktuellen Prozessorbeschreibungen (80386), neuen Musterprogrammen, nützlichen Routinen oder auch Erweiterungen der Makrobibliothek.

Mit zusätzlichem Diskettenservice!

Fordern Sie noch heute an:

### *Intel 16/32 Bit Assemblerhandbuch*

stabiler Ringbuchordner, DIN A 4, ca. 1100 Seiten, inkl. Diskette, Bestell-Nr. 2200/72200, Preis: DM 98,–.

Alle 2 – 3 Monate erhalten Sie Ergänzungsausgaben zum Grundwerk mit je ca. 120 Seiten zum Seitenpreis von 38 Pfennig (Abbestellung jederzeit möglich).

## Jetzt nutzen Sie Ihren PC noch effizienter

inklusive Diskette

Dieses Nachschlagewerk bietet Ihnen:

■ **detailliertes Hardwarewissen** über PCs, XTs, ATs bis hin zu den neuen PS/2-Systemen. Umfassende Prozessorbeschreibungen der Intelserie 80(x)xx zeigen Ihnen die CPU-Architekturen und -Beschaltungen sowie die internen Abläufe bei Adressierung und Befehlsausführung;

■ **Anleitungen für den routinierten Einsatz von Systemsoftware:** Sie lernen Aufbau und Funktionen von MS-/PC-DOS und Unix kennen und erfahren, welche Möglichkeiten Ihnen die System- und Benutzerschnittstellen bieten. Sie erfahren u.a. wie Sie das Betriebssystem Ihren individuellen Wünschen anpassen können oder wie Sie durch Emulieren von IBM-BIOS-Routinen auf PC-DOS abgestimmte Programme auch auf MS-DOS-Rechnern lauffähig machen;

■ **Tuning durch Hardwareerweiterungen,** u.a. Anleitung für den Festplatteneinbau; Einsatz von Coprozessoren sowie Bauanleitungen, z.B. für eine parallele I/O-Karte oder einen universellen EPROMer;

■ **neue Tools und Utilities:** u.a. MS-DOS-Utilities, Zweifach-Tastaturtreiber, Mausanschluß für Turbo-Pascal, speicherresidente Umwandlung des Druckercodes, Batchdateien, RAM-Disk, Turbo-Pascal-Schnittstelle zu MS-DOS, V.24-Treiber, Profiler, effizientes Arbeiten mit Standardsoftware (Symphony);

■ **den kompletten Assembler SIGMASM** für hardwarenahes Programmieren;

■ **Softwareengineering:** Sie eignen sich die optimale Vorgehensweise bei der Programmentwicklung an und lernen z.B. anspruchsvolle Beschreibungsmethoden wie SADT und HIPO-Verfahren zu nutzen.
Für komplexe Aufgabenstellungen erhalten Sie ausführlich dokumentierte Programmstrukturpläne. Bei der Arbeit mit Standardsoftware unterstützen Sie Templates;

■ **Musterlösungen,** z.B. zur Lagerverwaltung;

■ **Know-how für spezielle PC-Bereiche** wie DFÜ oder Netzwerke;

■ **Ergänzungsausgaben zum Grundwerk** mit neuer Anwendersoftware, weiteren Programmiersprachkursen wie C oder Lisp, aktuellen Hardware- und Betriebssystembeschreibungen (u.a. PCs mit 80386-Prozessor) sowie Tips und Utilities;

Zusätzlicher Diskettenservice!

Fordern Sie noch heute an:

### *Hard- und Softwaretuning für IBM PCs und Kompatible*

stabiler Ringbuchordner, DIN A 4, ca. 870 Seiten, inkl. Diskette, Bestell-Nr. 2600/72600, Preis: DM 98,–.

Alle 2 – 3 Monate erhalten Sie Ergänzungsausgaben zum Grundwerk mit je ca. 120 Seiten zum Seitenpreis von 38 Pfennig (Abbestellung jederzeit möglich).

## Basic-Tools, Utilities und Anwenderprogramme für Wirtschaft, Technik, Mathematik

inklusive Diskette

Dieses Nachschlagewerk bietet Ihnen:
auf ca. 1100 Seiten sofort einsetzbare Programme und Utilities für PC-/MS-DOS-Computer. Hinweise auf gerätespezifische Besonderheiten ermöglichen Ihnen auch die problemlose Anpassung an andere Rechner. Sie erhalten u.a.

■ **anspruchsvolle Anwenderprogramme wie**

– **die Textverarbeitung „SOTEXT 4.0";**

– **Programme für Wirtschaft und Verwaltung:** Finanzbuchhaltung „SOBUCH", Netzplantechnik, Terminkalender, Berechnung der Einkommenssteuer, Rabattermittlung, Tilgungsplan eines Darlehens, Restwertberechnung…;

– **mathematische Programme:**
Analyse von Funktionsgleichungen (z.B. Kurvendiskussion);

– **statistische Programme**
(u.a. Berechnung von Mittelwert, Streuung, Standardabweichung);

– **technische Programme:** Messen/Steuern/Regeln, Wertetabellen einer einfachen Digitalschaltung, Berechnen von Querschnittswerten…;

– **Programme für die Ausbildung** (z.B. Vokabeltraining);

■ **Tools und vielseitige Routinen für**

– **Grafik:** Grafik-Generator „SOGRAF", Zeichengenerator;

– **Dateiverwaltung und Dateiorganisation:** Dateigenerator, Komprimieren und Expandieren von Textdateien, Konvertieren von Dateien, Disketten-Editor „SOZAP", indexsequentielle Dateiorganisation und Reorganisation, Hash-Dateiorganisation; Such- und Sortierverfahren;

– **Drucker** (z.B. Ausgabe von Random-Dateien mit Zeilenvorschub, Druckereinstellung);

– **Bildschirm** (z.B. Bildschirm- und Menütechnik);

– **Multitasking** (z.B. Programm zur Parallelverarbeitung);

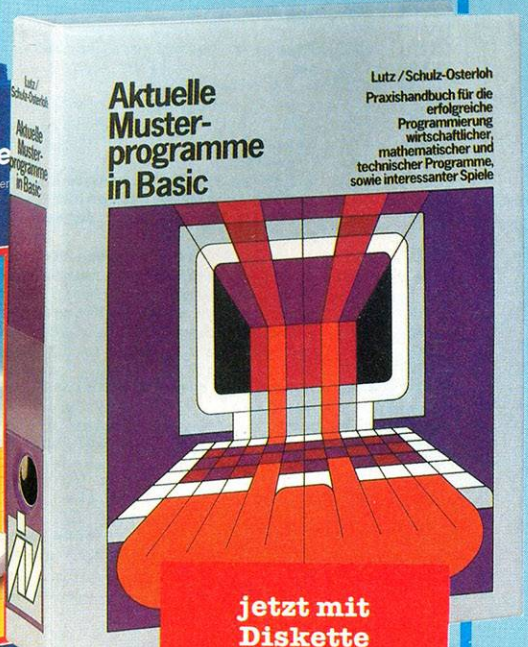
Jedes Programm ist ausführlich dokumentiert mit Problembeschreibung, Vorgehensweise, Struktogramm, Listing und Probelauf.

Fordern Sie noch heute an:

### *Aktuelle Musterprogramme in Basic*

stabiler Ringbuchordner, DIN A 4, über 1100 Seiten, inkl. Diskette „SOTEXT 4.0", Bestell-Nr. 1300/71300, Preis: DM 98,–.

Alle 2 – 3 Monate erhalten Sie Ergänzungsausgaben zum Grundwerk mit je ca. 120 Seiten zum Seitenpreis von 38 Pfennig (Abbestellung jederzeit möglich).





INTEREST VERLAG
Fachverlag für Special Interest Publikationen und Anwendersoftware

Industriestraße 21
D-8901 Kissing
Tel. 08233/21 01-0

```
[66M]  414:         =(seite.daten[var1,xpos shr 3,ypos s
       hr 2] or p) then test:=true
[9A5]  415:
                 else test:=false;
[E96]  416:    end else test:=false;
[TUO]  417: end;
[000]  418:
[83E]  419: procedure writetext(s:str255;xp,yp:integer)
       ;
[000]  420:
[EMO]  421: var
[L15]  422:    wx,wz : integer;
[X64]  423:    wn    : integer;
[HT4]  424:    ab,eb : integer;
[793]  425:    ak,ek : byte;
[6N2]  426:    ww    : byte;
[SK2]  427:    wc    : char;
[000]  428:
[068]  429: procedure bin(ww,j:byte);
[EMO]  430: var
[R92]  431:    w1 : byte;
[UA3]  432:    i  : integer;
[BH1]  433: begin
[ZZ3]  434:    w1:=ww shl (8-j);
[176]  435:    for i:=1 to j do begin
[LGD]  436:      if (w1 xor $80)<>(w1 or $80) then setpo
       int(wx,wz+yp)
[SB8]  437:                                   else reset
       point(wx,wz+yp);
[ZK2]  438:      w1:=w1 shl 1;
[4S3]  439:      wx:=succ(wx);
[TUO]  440:    end;
[TUO]  441: end;
[000]  442:
[BH1]  443: begin
[V6A]  444:    for wz:=1 to font.zeilen do begin
[OCG]  445:      if kursiv then wx:=xp+(font.zeilen-wz)
       shr 1 else wx:=xp;
[B99]  446:      for wn:=1 to length(s) do begin
[DOA]  447:        ab:=(font.daten[24+ord(s[wn]) shl 1]
[NAB]  448:             +256*font.daten[25+ord(s[wn]) sh
       l 1]) shr 3
[8V6]  449:             +font.offset*pred(wz);
[NOA]  450:        eb:=(font.daten[26+ord(s[wn]) shl 1]
[TAB]  451:             +256*font.daten[27+ord(s[wn]) sh
       l 1]) shr 3
[8V6]  452:             +font.offset*pred(wz);
[DXA]  453:        ak:=8-((font.daten[24+ord(s[wn]) shl
       1]
[RRB]  454:               +256*font.daten[25+ord(s[wn])
       shl 1]) mod 8);
[71A]  455:        ek:=(font.daten[26+ord(s[wn]) shl 1]
[WGB]  456:             +256*font.daten[27+ord(s[wn]) sh
       l 1]) mod 8;
[B96]  457:        while ab<=eb do begin
[6A7]  458:          ww:=font.daten[ab+$21a];
[Z85]  459:          if ab=eb then begin
[9X2]  460:            ak:=ek-8+ak;
[234]  461:            ww:=ww shr (8-ek)
[TUO]  462:          end;
[YHL]  463:          if uline and (wz>font.zeilen-((font
       .zeilen shr 3) or 1)) then ww:=255;
[XP2]  464:          bin(ww,ak);
[MK3]  465:          ab:=succ(ab);
[LA1]  466:          ak:=8;
[TUO]  467:        end;
[TUO]  468:      end;
[TUO]  469:    end;
[TUO]  470: end;
[000]  471:
[21C]  472: function textsize(s:str255):integer;
[EMO]  473: var
[BG3]  474:    ts : integer;
[4G3]  475:    wn : integer;
[000]  476:
[BH1]  477: begin
[EA1]  478:    ts:=0;
[PN6]  479:    for wn:=1 to length(s) do
[X4B]  480:      ts:=ts+font.daten[26+ord(s[wn]) shl 1]
[WU9]  481:           +256*font.daten[27+ord(s[wn]) shl
       1]
[B09]  482:           -font.daten[24+ord(s[wn]) shl 1]
[14A]  483:           -256*font.daten[25+ord(s[wn]) shl
       1];
[P8C]  484:    if kursiv then ts:=ts+font.zeilen shr 1;
[4C4]  485:    textsize:=ts;
[TUO]  486: end;
[000]  487:
[PAH]  488: procedure clearwind(x1,y1,sx,sy:integer;fla
       g:boolean);
[EMO]  489: var
[LS9]  490:    zaehler,zaehler1,adr:integer;
[BH1]  491: begin
[5K9]  492:    for zaehler:=y1 to y1+sy do begin
[OBD]  493:      if odd(zaehler) then adr:=8152+zaehler*
       40+(x1 shr 3)
[CK8]  494:                      else adr:=     zaehler*
       40+(x1 shr 3);
[8MD]  495:      if flag then fillchar(video[adr],sx shr
       3,0)
[5HC]  496:      else for zaehler1:=adr to adr+(sx shr 3
       )-1 do
[NKC]  497:        video[zaehler1]:=video[zaehler1] xor
       255;
[TUO]  498:    end;
[ZB5]  499:    if flag then begin
[7HC]  500:      draw((x1 shr 3) shl 3,y1,((x1+sx) shr 3
       ) shl 3,y1,1);
[NFD]  501:      draw(((x1+sx) shr 3) shl 3,y1,((x1+sx)
       shr 3) shl 3,y1+sy,0);
[83F]  502:      draw(((x1+sx) shr 3) shl 3,y1+sy,(x1 sh
       r 3) shl 3,y1+sy,1);
[DUB]  503:      draw((x1 shr 3) shl 3,y1+sy,(x1 shr 3)
       shl 3,y1,1);
[TUO]  504:    end;
[TUO]  505: end;
[000]  506:
[M4M]  507: function alert(str1,str2,str3:str40;but1,bu
       t2,but3:str20):byte;
[EMO]  508: var
[CW3]  509:    l1,l2,l3:byte;
[8V5]  510:    xpos,ypos:integer;
[GK3]  511:    alertw:byte;
[000]  512:
[BH1]  513: begin
[CE3]  514:    savescreen;
[E03]  515:    mouseoff;
[PZ4]  516:    l1:=length(str1);
[ZZ4]  517:    l2:=length(str2);
[805]  518:    l3:=length(str3);
[WH5]  519:    if l2>l1 then l1:=l2;
[5K5]  520:    if l3>l1 then l1:=l3;
[CY4]  521:    l2:=length(but2);
[NY4]  522:    l3:=length(but3);
[B8C]  523:    if length(but1)>l2 then l2:=length(but1);
[5K5]  524:    if l3>l2 then l2:=l3;
[R6B]  525:    clearwind(320-((l1+l2+5)*4),84,(l1+l2+5)*
       8,32,true);
[NMB]  526:    gotoxy(40-(l1+l2+1) div 2,12);write(str1)
       ;
[TMB]  527:    gotoxy(40-(l1+l2+1) div 2,13);write(str2)
       ;
[XMB]  528:    gotoxy(40-(l1+l2+1) div 2,14);write(str3)
       ;
[PRB]  529:    gotoxy(42-(l1+l2+1) div 2+l1,12);write(bu
       t1);
[OSB]  530:    gotoxy(42-(l1+l2+1) div 2+l1,13);write(bu
       t2);
[ASB]  531:    gotoxy(42-(l1+l2+1) div 2+l1,14);write(bu
       t3);
[9HB]  532:    draw(328-(l1+l2+2)*4+l1*8,87,328-(l1+l2+2
       )*4+l1*8,111,1);
[F6A]  533:    draw(328-(l1+l2+2)*4+l1*8,95,320+(l1+l2+2
       )*4,95,1);
[FKA]  534:    draw(328-(l1+l2+2)*4+l1*8,103,320+(l1+l2+
       2)*4,103,1);
[7KA]  535:    draw(328-(l1+l2+2)*4+l1*8,111,320+(l1+l2+
       2)*4,111,1);
[W6A]  536:    draw(328-(l1+l2+2)*4+l1*8,87,320+(l1+l2+2
       )*4,87,1);
[4C9]  537:    draw(320+(l1+l2+2)*4,87,320+(l1+l2+2)*4,1
       11,1);
[VGB]  538:    clearwind(320-((l1+l2+5)*4),84,(l1+l2+5)*
       8,32,false);
[B03]  539:    mouseon;
[UV2]  540:    alertw:=0;
[X52]  541:    repeat
[W96]  542:      repeat until lbutton;
[D28]  543:      repeat until not lbutton;
[PX3]  544:      xpos:=xmouse;
[YX3]  545:      ypos:=ymouse;
[EOE]  546:      if (xpos>(328-(l1+l2+2)*4+l1*8)) and (x
       pos<(320+(l1+l2+2)*4)) then
[GH3]  547:        case ypos of
[ML4]  548:          87..94  :alertw:=1;
[HN4]  549:          95..102 :alertw:=2;
[ON4]  550:          103..111:alertw:=3;
[TUO]  551:        end;
[8K4]  552:    until alertw>0;
[CC4]  553:    alert:=alertw;
[YC3]  554:    loadscreen;
[TUO]  555: end;
[000]  556:
[MZ7]  557: function readtext:str70;
[EMO]  558: var
[3N2]  559:    txt : str70;
[T42]  560:    bl  : byte;
[BH1]  561: begin
[E03]  562:    mouseoff;
[UC3]  563:    gotoxy(7,24);
[NH2]  564:    write(': ');
[023]  565:    bl:=buflen;
[TW2]  566:    buflen:=70;
[PU2]  567:    read(txt);
[W13]  568:    buflen:=bl;
[UC3]  569:    gotoxy(7,24);
```

```
[2W4]  570:    write(leerstr);
[FV4]  571:    readtext:=txt;
[BO3]  572:    mouseon;
[TUO]  573: end;
[OOO]  574:
[UX8]  575: procedure notexist(nr:byte);
[EMO]  576: var
[6E4]  577:    nes : string[3];
[UA2]  578:    nea : byte;
[BH1]  579: begin
[ZW2]  580:    case nr of
[8H3]  581:      1 :nes:='DTP';
[BH3]  582:      2 :nes:='FNT';
[VF3]  583:      3 :nes:='IMG';
[7L3]  584:      4 :nes:='TXT';
[TUO]  585:    end;
[RWG]  586:    nea:=alert('Die angegebene '+nes+'-Datei
       konnte nicht',
[E1D]  587:               'gefunden werden. Um weiterzua
       rbeiten,',
[PS9]  588:               'klicken Sie bitte OK an.','OK
       ','','');
[TUO]  589: end;
[OOO]  590:
[V98]  591: procedure speichervoll;
[EMO]  592: var
[1C2]  593:    sva : byte;
[BH1]  594: begin
[AUA]  595:    sva:=alert('Der Speicher ist voll !',
[HMA]  596:               'Um neue Blöcke zu definierten
       ',
[SKD]  597:               'löschen Sie bitte alte Blöcke
       .','WEITER','','');
[TUO]  598: end;
[OOO]  599:
[OOO]  600:
[EMO]  601: var
[5F6]  602:    katalog : inhalttyp;
[CC3]  603:    i       : byte;
[OOO]  604:
[CVC]  605: function getfilename(maske:str5):str70;
[EMO]  606: var
[WH3]  607:    pfad    : str64;
[9A4]  608:    akpfad  : str64;
[5K3]  609:    name    : str70;
[OOO]  610:
[NDF]  611: procedure lesedir(var kat:inhalttyp;msk:str
       64);
[EMO]  612: var
[Y54]  613:    dta   : dtatyp;
[EG4]  614:    regs  : regtyp;
[3O3]  615:    z1,z2 : byte;
[HY2]  616:    p1,p2 : byte;
[X13]  617:    msk1  : str64;
[2U3]  618:    laenge: real;
[OOO]  619:
[F9G]  620: procedure sort(var kat:inhalttyp;z3:byte;z4
       :byte);
[EMO]  621: var
[HH4]  622:    st    : string[13];
[ZO3]  623:    z5,z6 : byte;
[BH1]  624: begin
[R16]  625:    for z6:=z3 to pred(z4) do
[H16]  626:      for z5:=z3 to pred(z4) do
[5NC]  627:        if kat.name[z5]>kat.name[succ(z5)] th
       en
[BH1]  628:        begin
[275]  629:          st:=kat.name[z5];
[ZG9]  630:          kat.name[z5]:=kat.name[succ(z5)];
[2H7]  631:          kat.name[succ(z5)]:=st;
[TUO]  632:        end;
[TUO]  633: end;
[OOO]  634:
[RGF]  635: function dta_nach_str(name:str13;z:char):st
       r13;
[EMO]  636: var
[HY2]  637:    p1,p2 : byte;
[5Z2]  638:    dns   : str13;
[BH1]  639: begin
[B92]  640:    dns:=z;
[B61]  641:    p1:=1;
[DF4]  642:    if name[1]='.' then
[B53]  643:      dns:=#04+name
[A11]  644:    else
[3O4]  645:      for p2:=2 to 13 do
[3Y7]  646:        if (p2=10) and (name[p1]<>'.') then
[BX2]  647:          dns:=dns+'.'
[A11]  648:        else
[CSC]  649:          if ((name[p1]='.') and (p2<10)) or
       (name[p1]=#0) then
[Y13]  650:            dns:=dns+'_'
[A11]  651:          else
[BH1]  652:          begin
[8F5]  653:            dns:=dns+name[p1];
[CC3]  654:            p1:=succ(p1);
[TUO]  655:          end;
[AD5]  656:    dta_nach_str:=dns;
[TUO]  657: end;
[OOO]  658:
```

```
[BH1]  659: begin
[GWF]  660:    for z1:=1 to length(msk) do if msk[z1]='\
       ' then z2:=z1;
[ST6]  661:    msk1:=copy(msk,1,z2)+'*.*'+#0;
[N61]  662:    z1:=1;
[HR6]  663:    fillchar(dta.name,13,#32);
[FF3]  664:    with regs do
[BH1]  665:    begin
[TPB]  666:      ah:=$0e; dx:=ord(msk[1])-65; msdos(regs
       );
[HBD]  667:      ah:=$1a; ds:=seg(dta); dx:=ofs(dta); ms
       dos(regs);
[DEH]  668:      ah:=$4e; ds:=seg(msk1[1]); dx:=ofs(msk1
       [1]); cx:=$10; msdos(regs);
[TUO]  669:    end;
[MY5]  670:    while (regs.fl and 1)=0 do
[BH1]  671:    begin
[GU7]  672:      if dta.attribut=16 then
[BH1]  673:      begin
[VBB]  674:        kat.name[z1]:=dta_nach_str(dta.name,#
       4);
[AD3]  675:        z1:=succ(z1);
[TUO]  676:      end;
[HR6]  677:      fillchar(dta.name,13,#32);
[Z87]  678:      regs.ah:=$4f; msdos(regs);
[TUO]  679:    end;
[5X1]  680:    z2:=z1;
[GX5]  681:    sort(kat,1,pred(z1));
[MZ2]  682:    msk1:=msk+#0;
[HR6]  683:    fillchar(dta.name,13,#32);
[FF3]  684:    with regs do
[BH1]  685:    begin
[FXH]  686:      ah:=$4e; ds:=seg(msk1[1]); dx:=ofs(msk1
       [1]); cx:=37; msdos(regs);
[TUO]  687:    end;
[MY5]  688:    while (regs.fl and 1)=0 do
[BH1]  689:    begin
[B61]  690:      p1:=1;
[FFB]  691:      kat.name[z1]:=dta_nach_str(dta.name,#32
       );
[AD3]  692:      z1:=succ(z1);
[HR6]  693:      fillchar(dta.name,13,#32);
[Z87]  694:      regs.ah:=$4f; msdos(regs);
[TUO]  695:    end;
[F76]  696:    sort(kat,z2,pred(z1));
[GF5]  697:    kat.anz:=pred(z1);
[TUO]  698: end;
[OOO]  699:
[256]  700: procedure zeigebox;
[EMO]  701: var
[U92]  702:    z1 : byte;
[BH1]  703: begin
[E03]  704:    mouseoff;
[4D7]  705:    clearwind(248,44,136,88,true);
[BP4]  706:    draw(368,44,368,132,1);
[VA4]  707:    draw(368,52,383,52,1);
[UK4]  708:    draw(368,124,383,124,1);
[ZR3]  709:    for z1:=0 to 8 do
[BH1]  710:    begin
[DS4]  711:      draw(376,46,372+z1,50,1);
[935]  712:      draw(376,130,372+z1,126,1);
[TUO]  713:    end;
[EZ7]  714:    clearwind(248,44,136,88,false);
[BO3]  715:    mouseon;
[TUO]  716: end;
[OOO]  717:
[LTF]  718: procedure zeigezeiger(kat:inhalttyp;st,fa:b
       yte);
[EMO]  719: var
[8T3]  720:    hoehe  : byte;
[E84]  721:    stelle : byte;
[BH1]  722: begin
[VC4]  723:    if kat.anz<=10 then
[BH1]  724:    begin
[VD2]  725:      hoehe:=70;
[PW2]  726:      stelle:=53;
[LM2]  727:    end else
[BH1]  728:    begin
[EV6]  729:      hoehe:=trunc(700/kat.anz);
[MND]  730:      stelle:=trunc((71-hoehe)/(kat.anz-10)*p
       red(st))+53;
[TUO]  731:    end;
[E03]  732:    mouseoff;
[4VA]  733:    draw(370,stelle,370,stelle+hoehe,fa);
[8DC]  734:    draw(370,stelle+hoehe,382,stelle+hoehe,fa
       );
[VNA]  735:    draw(382,stelle+hoehe,382,stelle,fa);
[R49]  736:    draw(382,stelle,370,stelle,fa);
[BO3]  737:    mouseon;
[TUO]  738: end;
[OOO]  739:
[24E]  740: procedure zeigedir(kat:inhalttyp;st:byte);
[EMO]  741: var
[U92]  742:    z1 : byte;
[BH1]  743: begin
[E03]  744:    mouseoff;
[5S3]  745:    for z1:=0 to 9 do
[BH1]  746:    begin
[694]  747:      gotoxy(33,7+z1);
```

```
[3W6]  748:      if (z1+st)<=kat.anz then
[DU6]  749:        write(kat.name[z1+st])
[A11]  750:      else
[ZC6]  751:        write('  ________.___');
[TUO]  752:    end;
[SY7]  753:    clearwind(256,48,104,79,false);
[BO3]  754:    mouseon;
[HT6]  755:    zeigezeiger(kat.st,0);
[TUO]  756: end;
[OOO]  757:
[41E]  758: function mouseauswahl(kat:inhalttyp):str70;
[EMO]  759: var
[WM2]  760:    st     : byte;
[XG2]  761:    z1     : byte;
[5Z4]  762:    xm,ym  : integer;
[UK2]  763:    n      : str13;
[FE3]  764:    hoehe  : byte;
[EG4]  765:    regs   : regtyp;
[OOO]  766:
[BH1]  767: begin
[MA1]  768:    st:=1;
[X52]  769:    repeat
[1FC]  770:      if (copy(kat.name[1],2,1)<>'.') or (kat
           .anz=0) then
[BH1]  771:        begin
[UW7]  772:          for z1:=kat.anz downto 1 do
[T88]  773:            kat.name[z1+2]:=kat.name[z1];
[X66]  774:          kat.name[1]:=#4+'.          ';
[986]  775:          kat.name[2]:=#4+'..         ';
[4M5]  776:          kat.anz:=kat.anz+2;
[TUO]  777:        end;
[HH5]  778:      zeigedir(kat,st);
[X52]  779:      repeat
[U21]  780:        n:='';
[KNA]  781:        repeat until lbutton or rbutton;
[MG5]  782:        if rbutton then
[BH1]  783:        begin
[T28]  784:          repeat until not rbutton;
[U71]  785:          n:='.';
[LM2]  786:        end else
[BH1]  787:        begin
[H63]  788:          xm:=xmouse;
[R63]  789:          ym:=ymouse;
[G2E]  790:          if (xm>367) and (xm<385) and (ym>43
           ) and (ym<133) then
[BH1]  791:            begin
[NT6]  792:              zeigezeiger(kat.st,1);
[FBD]  793:              if (ym>43) and (ym<53) and (st>1)
            then st:=pred(st);
[6RG]  794:              if (ym>123) and (ym<133) and (st<
           (kat.anz-9)) then st:=succ(st);
[ZP6]  795:              if (ym>52) and (ym<124) then
[BH1]  796:              begin
[92H]  797:                if kat.anz<=10 then hoehe:=70 e
           lse hoehe:=trunc(700/kat.anz);
[2D8]  798:                mousewindow(0,53,639,124-hoehe)
           ;
[X52]  799:                repeat
[R63]  800:                  ym:=ymouse;
[NT6]  801:                  zeigezeiger(kat.st,1);
[6W7]  802:                  if kat.anz<11 then st:=1 else
[1RC]  803:                    st:=succ(trunc((ym-53)*(kat
           .anz-10)/(71-hoehe)));
[HT6]  804:                  zeigezeiger(kat.st,0);
[OP5]  805:                until not lbutton;
[L56]  806:                mousewindow(0,0,639,199);
[TUO]  807:              end;
[B41]  808:              n:=#4;
[TUO]  809:            end;
[Y2E]  810:            if (xm>248) and (xm<368) and (ym>47
           ) and (ym<128) then
[BH1]  811:            begin
[2U9]  812:              if (st+ (ym-48) div 8)>kat.anz th
           en n:=''
[FG9]  813:                                          el
           se n:=kat.name[st+ (ym-48) div 8];
[V2E]  814:              for z1:=1 to length(n) do if n[z1
           ]='_' then n[z1]:=#32;
[PUD]  815:              if (n[11]=#32) and (n[12]=#32) an
           d (n[13]=#32) then n:=copy(n,1,9);
[8R4]  816:              if n[1]=#4 then
[TZ8]  817:                if (n[2]='.') and (n[3]<>'.') t
           hen n:='  .'
[YS2]  818:                else begin
[ASA]  819:                  if (n[3]='.') and (length(pfa
           d)=3) then
[BH1]  820:                  begin
[NT6]  821:                    zeigezeiger(kat,st,1);
[DO4]  822:                    intr($11,regs);
[NZ7]  823:                    kat.anz:=2+(regs.al shr 6);
[D89]  824:                    kat.name[1]:=#4+'.
            '; (*11 spaces*)
[EX6]  825:                    for z1:=2 to kat.anz do
[ZOC]  826:                      kat.name[z1]:=#4+chr(63+z
           1)+'           '; (*10 spaces*)
[DW6]  827:                    if setup.drive>0 then
[BH1]  828:                    begin
[4V7]  829:                      for z1:=1 to setup.drive
           do
[PKE]  830:                        kat.name[kat.anz+z1]:=#
           4+chr(66+z1)+'           '; (*10 spaces*)
[O79]  831:                      kat.anz:=kat.anz+setup.dr
           ive;
[TUO]  832:                    end;
[MA1]  833:                    st:=1;
[LM2]  834:                  end else
[BH1]  835:                  begin
[NT6]  836:                    zeigezeiger(kat,st,1);
[S3D]  837:                    while n[length(n)]=#32 do d
           elete(n,length(n),1);
[VM4]  838:                    chdir(copy(n,2,12));
[RH4]  839:                    getdir(0,pfad);
[AMD]  840:                    if pfad[length(pfad)]<>'\'
           then pfad:=pfad+'\';
[7X7]  841:                    lesedir(kat,pfad+maske);
[MA1]  842:                    st:=1;
[TUO]  843:                  end;
[TUO]  844:                end;
[TUO]  845:              end;
[D28]  846:              repeat until not lbutton;
[TUO]  847:            end;
[UV4]  848:        until length(n)>0;
[A24]  849:      until n[1]<>#4;
[5K9]  850:      mouseauswahl:=pfad+copy(n,2,12);
[TUO]  851: end;
[OOO]  852:
[BH1]  853: begin
[CE3]  854:    savescreen;
[S85]  855:    getdir(0,akpfad);
[ZFF]  856:    if akpfad[length(akpfad)]='\' then pfad:=
           akpfad
[CC6]  857:                                         else pfad:=
           akpfad+'\';
[PY8]  858:    lesedir(katalog,pfad+maske);
[4X2]  859:    zeigebox;
[CT8]  860:    name:=mouseauswahl(katalog);
[KZ3]  861:    chdir(akpfad);
[YC3]  862:    loadscreen;
[EO3]  863:    mouseoff;
[UC3]  864:    gotoxy(7,24);
[F73]  865:    write(name);
[BO3]  866:    mouseon;
[PG5]  867:    getfilename:=name;
[TUO]  868: end;
[OOO]  869:
[U6B]  870: procedure loadfont(fontname : str70);
[EMO]  871: var
[D54]  872:    fz       : integer;
[8Y7]  873:    kanal    : file of datentyp;
[CP3]  874:    xp       : byte;
[BH1]  875: begin
[EO3]  876:    mouseoff;
[UC3]  877:    gotoxy(7,24);
[2W4]  878:    write(leerstr);
[Z77]  879:    assign(kanal,fontname);
[3R6]  880:    {$I-} reset(kanal); {$I+}
[D45]  881:    if ioresult=0 then
[AF7]  882:      if filesize(kanal)>1 then
[BH1]  883:      begin
[BO3]  884:        mouseon;
[YKB]  885:        fz:=alert('Die FNT-Datei ist zu groß'
           ,
[ZX7]  886:                  'und kann deshalb nicht',
[HX7]  887:                  'geladen werden.','OK','','
           ');
[LM2]  888:      end else
[BH1]  889:      begin
[FF7]  890:        read(kanal,font.daten);
[UVD]  891:        font.offset:=font.daten[$50]+256*font
           .daten[$51];
[GXD]  892:        font.zeilen:=font.daten[$52]+256*font
           .daten[$53];
[PY4]  893:        font.name:='';fz:=4;
[OML]  894:        for fz:=4 to 8 do font.name:=concat(f
           ont.name,chr(font.daten[fz]));
[ML6]  895:        font.name:=font.name+
[O3H]  896:                   '('+chr(48+font.daten[2] d
           iv 10)+chr(48+font.daten[2] mod 10)+')';
[UC3]  897:        gotoxy(1,24);
[295]  898:        write(font.name);
[HA1]  899:        xp:=1;
[X52]  900:        repeat
[ZP3]  901:          xp:=succ(xp);
[3E5]  902:          gotoxy(pred(xp),24);
[HG5]  903:          write(#32,font.name);
[BN5]  904:        until xp=fontpos;
[KK5]  905:        gotoxy(fontpos,24);
[A83]  906:        write('          ');
[LK5]  907:        gotoxy(fontpos,25);
[295]  908:        write(font.name);
[BO3]  909:        mouseon;
[DLO]  910:      end
[A11]  911:    else
[BH1]  912:    begin
[BO3]  913:      mouseon;
[2H3]  914:      notexist(2);
[TUO]  915:    end;
[9L3]  916:    close(kanal);
```

```
[TUO]  917: end;
[000]  918:
[889]  919: procedure drawbox(blnr:byte);
[BH1]  920: begin
[OKB]  921:   xordraw(seite.block[blnr].xpos shr 1,
[6K8]  922:           seite.block[blnr].ypos shr 1,
[Y5K]  923:           (seite.block[blnr].xpos+seite.blo
       ck[blnr].breite) shr 1,
[TW8]  924:           seite.block[blnr].ypos shr 1);
[V5N]  925:   xordraw((seite.block[blnr].xpos+seite.blo
       ck[blnr].breite) shr 1,
[6K8]  926:           seite.block[blnr].ypos shr 1,
[Y5K]  927:           (seite.block[blnr].xpos+seite.blo
       ck[blnr].breite) shr 1,
[HWH]  928:           (seite.block[blnr].ypos+seite.blo
       ck[blnr].hoehe) shr 1);
[V5N]  929:   xordraw((seite.block[blnr].xpos+seite.blo
       ck[blnr].breite) shr 1,
[6XH]  930:           (seite.block[blnr].ypos+seite.blo
       ck[blnr].hoehe) shr 1,
[3K8]  931:           seite.block[blnr].xpos shr 1,
[HWH]  932:           (seite.block[blnr].ypos+seite.blo
       ck[blnr].hoehe) shr 1);
[OKB]  933:   xordraw(seite.block[blnr].xpos shr 1,
[6XH]  934:           (seite.block[blnr].ypos+seite.blo
       ck[blnr].hoehe) shr 1,
[3K8]  935:           seite.block[blnr].xpos shr 1,
[TW8]  936:           seite.block[blnr].ypos shr 1);
[TUO]  937: end;
[000]  938:
[DSB]  939: function movebox(blnr:byte):boolean;
[EMO]  940: var
[WV5]  941:   xb,yb             : integer;
[UP8]  942:   xm,xmanf,ym,ymanf : integer;
[000]  943:
[BH1]  944: begin
[RX4]  945:   movebox:=true;
[E03]  946:   mouseoff;
[22F]  947:   mousewindow(0,0,319-(seite.block[blnr].br
       eite shr 1),
[UCA]  948:                    171-(seite.block[blnr].ho
       ehe shr 1));
[UE8]  949:   xb:=seite.block[blnr].xpos;
[2F8]  950:   yb:=seite.block[blnr].ypos;
[Y84]  951:   xmanf:=xb shr 1;
[Z84]  952:   ymanf:=yb shr 1;
[W17]  953:   setmouse(xmanf,ymanf);
[X52]  954:   repeat
[EM4]  955:     drawbox(blnr);
[X52]  956:     repeat
[H63]  957:       xm:=xmouse;
[R63]  958:       ym:=ymouse;
[U5H]  959:     until (xm<>xmanf) or (ym<>ymanf) or lbu
       tton or rbutton;
[EM4]  960:     drawbox(blnr);
[P6A]  961:     seite.block[blnr].xpos:=xm shl 1;
[NG2]  962:     xmanf:=xm;
[U6A]  963:     seite.block[blnr].ypos:=ym shl 1;
[PG2]  964:     ymanf:=ym;
[368]  965:   until lbutton or rbutton;
[MG5]  966:   if rbutton then
[BH1]  967:   begin
[UC8]  968:     seite.block[blnr].xpos:=xb;
[YC8]  969:     seite.block[blnr].ypos:=yb;
[3H5]  970:     movebox:=false;
[TUO]  971:   end;
[L56]  972:   mousewindow(0,0,639,199);
[B03]  973:   mouseon;
[TUO]  974: end;
[000]  975:
[4VD]  976: function getblock(xpos,ypos:integer):byte;
[EMO]  977: var
[ER4]  978:   schleife:byte;
[BH1]  979: begin
[904]  980:   getblock:=255;
[R48]  981:   if seite.blocknr>0 then
[PYB]  982:     for schleife:=0 to pred(seite.blocknr)
       do
[1NC]  983:       if ((xpos shl 1)>seite.block[schleife
       ].xpos)
[PAR]  984:       and ((xpos shl 1 -seite.block[schleif
       e].breite)<seite.block[schleife].xpos)
[5XD]  985:       and ((ypos shl 1)>seite.block[schleif
       e].ypos)
[Z9E]  986:       and ((ypos shl 1 -seite.block[schleif
       e].hoehe)
[3F8]  987:           <seite.block[schleife].ypos)
[8L7]  988:       then getblock:=schleife;
[TUO]  989: end;
[000]  990:
[CZ9]  991: procedure delblock(blnr:byte);
[EMO]  992: var
[266]  993:   dschleife  : integer;
[476]  994:   dschleife1 : integer;
[RN5]  995:   htrans     : boolean;
[BH1]  996: begin
[X05]  997:   if blnr<255 then
[BH1]  998:   begin
[E03]  999:     mouseoff;
```

```
[CL4] 1000:     htrans:=trans;
[6P3] 1001:     trans:=false;
[YXX] 1002:     for dschleife:=seite.block[blnr].ypos t
       o (seite.block[blnr].ypos+seite.block[blnr].
       hoehe) do
[9FY] 1003:       for dschleife1:=seite.block[blnr].xpo
       s to (seite.block[blnr].xpos+seite.block[bln
       r].breite) do
[DSA] 1004:         resetpoint(dschleife1,dschleife);
[1EA] 1005:     if seite.blocknr>succ(blnr) then
[YND] 1006:       for dschleife:=blnr to pred(seite.blo
       cknr) do
[BH1] 1007:       begin
[3SM] 1008:         seite.block[dschleife].xpos:=seite.
       block[succ(dschleife)].xpos;
[6SM] 1009:         seite.block[dschleife].ypos:=seite.
       block[succ(dschleife)].ypos;
[67N] 1010:         seite.block[dschleife].breite:=seit
       e.block[succ(dschleife)].breite;
[9PM] 1011:         seite.block[dschleife].hoehe:=seite
       .block[succ(dschleife)].hoehe;
[TUO] 1012:       end;
[DA4] 1013:     trans:=htrans;
[CLA] 1014:     seite.blocknr:=pred(seite.blocknr);
[B03] 1015:     mouseon;
[003] 1016:     display;
[TUO] 1017:   end;
[TUO] 1018: end;
[000] 1019:
[YHA] 1020: procedure copyblock(blnr:byte);
[EMO] 1021: var
[L25] 1022:   cs1,cs2 : integer;
[UN4] 1023:   cs,ce   : integer;
[BH1] 1024: begin
[X05] 1025:   if blnr<255 then
[BH1] 1026:   begin
[C3L] 1027:     seite.block[seite.blocknr].xpos:=seite.
       block[blnr].xpos;
[M3L] 1028:     seite.block[seite.blocknr].ypos:=seite.
       block[blnr].ypos;
[WNL] 1029:     seite.block[seite.blocknr].breite:=seit
       e.block[blnr].breite;
[X4L] 1030:     seite.block[seite.blocknr].hoehe:=seite
       .block[blnr].hoehe;
[GCA] 1031:     if movebox(seite.blocknr) then
[BH1] 1032:     begin
[E03] 1033:       mouseoff;
[8LA] 1034:       seite.blocknr:=succ(seite.blocknr);
[FNP] 1035:       if seite.block[pred(seite.blocknr)].y
       pos>seite.block[blnr].ypos then
[BH1] 1036:       begin
[TS8] 1037:         cs2:=seite.block[blnr].hoehe;
[LL1] 1038:         ce:=-1;
[FM1] 1039:         cs:=-1;
[DLO] 1040:       end
[YS2] 1041:       else begin
[2PA] 1042:         ce:=succ(seite.block[blnr].hoehe);
[NN1] 1043:         cs2:=0;
[2A1] 1044:         cs:=1;
[TUO] 1045:       end;
[5G4] 1046:       while cs2<>ce do
[BH1] 1047:       begin
[OGC] 1048:         for cs1:=0 to seite.block[blnr].bre
       ite do
[1VM] 1049:           if test(cs1+seite.block[blnr].xpo
       s,cs2+seite.block[blnr].ypos) then
[G9X] 1050:             setpoint(cs1+seite.block[pred(s
       eite.blocknr)].xpos,cs2+seite.block[pred(sei
       te.blocknr)].ypos)
[A11] 1051:           else
[ENZ] 1052:             resetpoint(cs1+seite.block[pred
       (seite.blocknr)].xpos,cs2+seite.block[pred(s
       eite.blocknr)].ypos);
[553] 1053:         cs2:=cs2+cs;
[TUO] 1054:       end;
[B03] 1055:       mouseon;
[003] 1056:       display;
[TUO] 1057:     end;
[TUO] 1058:   end;
[TUO] 1059: end;
[000] 1060:
[2GA] 1061: procedure moveblock(blnr:byte);
[EMO] 1062: var
[L25] 1063:   cs1,cs2 : integer;
[UN4] 1064:   cs,ce   : integer;
[SC5] 1065:   htrans  : boolean;
[HR5] 1066:   sbx,sby : integer;
[SH5] 1067:   sbb,sbh : integer;
[BH1] 1068: begin
[X05] 1069:   if blnr<255 then
[BH1] 1070:   begin
[OK8] 1071:     sbx:=seite.block[blnr].xpos;
[3K8] 1072:     sby:=seite.block[blnr].ypos;
[DG9] 1073:     sbb:=seite.block[blnr].breite;
[9X8] 1074:     sbh:=seite.block[blnr].hoehe;
[217] 1075:     if movebox(blnr) then
[BH1] 1076:     begin
[E03] 1077:       mouseoff;
[MOB] 1078:       if seite.block[blnr].ypos>sby then
```

```
[BH1] 1079:         begin
[LC2] 1080:           cs2:=sbh;
[LL1] 1081:           ce:=-1;
[FM1] 1082:           cs:=-1;
[DLO] 1083:         end
[YS2] 1084:         else begin
[444] 1085:           ce:=succ(sbh);
[NN1] 1086:           cs2:=0;
[2A1] 1087:           cs:=1;
[TUO] 1088:         end;
[5G4] 1089:         while cs2<>ce do
[BH1] 1090:         begin
[TB5] 1091:           for cs1:=0 to sbb do
[PF9] 1092:             if test(cs1+sbx,cs2+sby) then
[BH1] 1093:             begin
[CL4] 1094:               htrans:=trans;
[6P3] 1095:               trans:=false;
[RN7] 1096:               resetpoint(cs1+sbx,cs2+sby);
[DA4] 1097:               trans:=htrans;
[2LM] 1098:               setpoint(cs1+seite.block[blnr].
      xpos,cs2+seite.block[blnr].ypos);
[LM2] 1099:             end else
[KEN] 1100:               resetpoint(cs1+seite.block[blnr
      ].xpos,cs2+seite.block[blnr].ypos);
[553] 1101:           cs2:=cs2+cs;
[TUO] 1102:         end;
[BO3] 1103:         mouseon;
[003] 1104:         display;
[TUO] 1105:       end;
[TUO] 1106:     end;
[TUO] 1107: end;
[000] 1108:
[C56] 1109: procedure dtpblock;
[EMO] 1110: var
[RSA] 1111:   xpos,ypos,breite,hoehe : integer;
[T79] 1112:   xm,xmanf,ym,ymanf       : integer;
[BH1] 1113: begin
[9HA] 1114:   if seite.blocknr<maxblock then
[BH1] 1115:   begin
[KNA] 1116:     repeat until lbutton or rbutton;
[PX3] 1117:     xpos:=xmouse;
[YX3] 1118:     ypos:=ymouse;
[D28] 1119:     repeat until not lbutton;
[GRD] 1120:     if not rbutton and (xpos<320) and (ypos
      <172) then
[BH1] 1121:     begin
[E03] 1122:       mouseoff;
[836] 1123:       mousewindow(0,0,319,171);
[563] 1124:       xmanf:=xpos;
[663] 1125:       ymanf:=ypos;
[XV4] 1126:       ypos:=ypos shl 1;
[NV4] 1127:       xpos:=xpos shl 1;
[X52] 1128:       repeat
[F87] 1129:         if xpos<(xmanf shl 1) then
[ZKC] 1130:           seite.block[seite.blocknr].xpos:=
      xpos
[A11] 1131:         else
[M6E] 1132:           seite.block[seite.blocknr].xpos:=
      xmanf shl 1;
[N87] 1133:         if ypos<(ymanf shl 1) then
[5LC] 1134:           seite.block[seite.blocknr].ypos:=
      ypos
[A11] 1135:         else
[U6E] 1136:           seite.block[seite.blocknr].ypos:=
      ymanf shl 1;
[FYH] 1137:         seite.block[seite.blocknr].breite:=
      abs(xpos-(xmanf shl 1));
[NVH] 1138:         seite.block[seite.blocknr].hoehe:=a
      bs(ypos-(ymanf shl 1));
[HS7] 1139:         drawbox(seite.blocknr);
[X52] 1140:         repeat
[H63] 1141:           xm:=xmouse;
[R63] 1142:           ym:=ymouse;
[U5H] 1143:         until (xm<>xmanf) or (ym<>ymanf) or
       lbutton or rbutton;
[HS7] 1144:         drawbox(seite.blocknr);
[NG2] 1145:         xmanf:=xm;
[PG2] 1146:         ymanf:=ym;
[368] 1147:       until lbutton or rbutton;
[L56] 1148:       mousewindow(0,0,639,199);
[BO3] 1149:       mouseon;
[RKH] 1150:       if not rbutton then seite.blocknr:=su
      cc(seite.blocknr);
[TUO] 1151:     end;
[DC7] 1152:   end else speichervoll;
[TUO] 1153: end;
[000] 1154:
[3W6] 1155: procedure bildaufbau;
[EMO] 1156: var
[1R5] 1157:   schleife:integer;
[BH1] 1158: begin
[5Y1] 1159:   hires;
[FU8] 1160:   fillchar(video[$0000],7360,$aa);
[DB8] 1161:   fillchar(video[$2000],7360,$55);
[N58] 1162:   fillchar(seite.daten,27520,0);
[XC3] 1163:   gotoxy(1,25);
[5S4] 1164:   write(prgname);
[9P3] 1165:   gotoxy(55,25);
[B35] 1166:   write('written by');
```

```
[7R6] 1167:   for schleife:=0 to 5 do
[BH1] 1168:   begin
[XR6] 1169:     gotoxy(43+6*schleife,1);
[397] 1170:     write(menu[schleife,0]);
[TUO] 1171:   end;
[A54] 1172:   draw(0,172,320,172,0);
[R64] 1173:   draw(320,0,320,172,0);
[X64] 1174:   draw(321,0,321,172,0);
[RB6] 1175:   for schleife:=0 to 85 do
[BH1] 1176:   begin
[X8A] 1177:     fillchar(video[schleife*80],40,255);
[Y4B] 1178:     fillchar(video[schleife*80+$2000],40,25
      5);
[TUO] 1179:   end;
[AP3] 1180:   gotoxy(66,25);
[8U9] 1181:   for schleife:=1 to system[0] do
[8RL] 1182:     write(chr((system[schleife] shl 1) or (
      system[schleife] shr 7)));
[L56] 1183:   mousewindow(0,0,639,199);
[BO3] 1184:   mouseon;
[1B3] 1185:   aktmenu:=6;
[TUO] 1186: end;
[000] 1187:
[AV5] 1188: procedure ldtxt;
[EMO] 1189: var
[W75] 1190:   setzart     : byte;
[CN5] 1191:   breite      : integer;
[Z55] 1192:   hoehe       : integer;
[OC5] 1193:   anzspalten  : byte;
[W85] 1194:   txtname     : str70;
[434] 1195:   kanal       : text;
[X86] 1196:   xp,yp,xz,yz: integer;
[WD4] 1197:   xp1          : integer;
[3R4] 1198:   ls,ls1,ls2 : str255;
[SR3] 1199:   ls3          : str255;
[SO5] 1200:   schleife    : byte;
[BH1] 1201: begin
[9HA] 1202:   if seite.blocknr<maxblock then
[BH1] 1203:   begin
[EP8] 1204:     txtname:=getfilename('*.TXT');
[UKB] 1205:     if txtname[length(txtname)]<>'.' then
[BH1] 1206:     begin
[TS6] 1207:       assign(kanal,txtname);
[3R6] 1208:       {$I-} reset(kanal); {$I+}
[D45] 1209:       if ioresult=0 then
[BH1] 1210:       begin
[ZF1] 1211:       ls3:='';
[X52] 1212:       repeat
[VA9] 1213:         schleife:=succ(seite.blocknr);
[AY2] 1214:         dtpblock;
[46A] 1215:         if schleife=seite.blocknr then
[BH1] 1216:         begin
[KBD] 1217:           xp:=seite.block[pred(seite.blockn
      r)].xpos;
[TBD] 1218:           yp:=seite.block[pred(seite.blockn
      r)].ypos;
[ESF] 1219:           breite:=seite.block[pred(seite.bl
      ocknr)].breite;
[15K] 1220:           hoehe:=seite.block[pred(seite.blo
      cknr)].hoehe-font.zeilen;
[N6D] 1221:           setzart:=alert('Wie soll der zu l
      adene',
[NN5] 1222:                                'Text dargestellt'
      ,
[URC] 1223:                                'werden ?','LINKSB
      ÜNDIG','ZENTRIERT','BLOCKSATZ');
[PEC] 1224:           anzspalten:=alert('In wieviele Sp
      alten',
[T67] 1225:                                  'soll der Text
      geteilt',
[865] 1226:                                  'werden ?','1',
      '2','3');
[XR9] 1227:           breite:=breite div anzspalten;
[X58] 1228:           xp1:=xp+breite*anzspalten;
[E03] 1229:           mouseoff;
[YH1] 1230:           ls:='';
[WF1] 1231:           ls2:='';
[OB1] 1232:           xz:=0;
[BXD] 1233:           while (not(eof(kanal)) or (length
      (ls)>0)) and (xp<xp1) do
[BH1] 1234:           begin
[1B1] 1235:             yz:=0;
[8DF] 1236:             while (not(eof(kanal)) or (leng
      th(ls)>0)) and (yz<hoehe) do
[BH1] 1237:             begin
[1H5] 1238:               if length(ls)=0 then
[V4B] 1239:                 if length(ls3)=0 then readl
      n(kanal,ls)
[AW8] 1240:                                      else begin
       ls:=ls3;ls3:='';end;
[TSA] 1241:               while (length(ls)>0) and (yz<
      hoehe) do
[BH1] 1242:               begin
[TF1] 1243:                 ls1:='';
[8GE] 1244:                 while (length(ls)>0) and (l
      s[1]=#32) do delete(ls,1,1);
[68D] 1245:                 while (ls1[length(ls1)]<>#3
      2) and (length(ls)>0) do
[BH1] 1246:                 begin
```

```
[AH6] 1247:                    ls1:=concat(ls1,ls[1]);
[PU3] 1248:                    delete(ls,1,1);
[TU0] 1249:                  end;
[NFB] 1250:                  if length(ls)=0 then ls1:=c
      oncat(ls1,#32);
[UAD] 1251:                  if ((xz+textsize(ls1))>brei
      te) and (xz>0) then
[BH1] 1252:                  begin
[B1B] 1253:                    if (setzart=3) and (lengt
      h(ls2)>1) then
[BH1] 1254:                    begin
[5B1] 1255:                      xz:=1;
[AC3] 1256:                      schleife:=0;
[G3H] 1257:                      while (textsize(concat(
      ls2,#32))<breite) and (schleife<2) do
[BH1] 1258:                      begin
[ZGG] 1259:                        while (ls2[xz]<>#32)
      and (xz<length(ls2)) do xz:=succ(xz);
[K67] 1260:                        if xz<length(ls2) the
      n
[BH1] 1261:                        begin
[DC3] 1262:                          schleife:=1;
[3T4] 1263:                          insert(#32,ls2,xz);
[NBD] 1264:                          if (xz+2)<length(ls
      2) then xz:=xz+2 else xz:=1;
[SRE] 1265:                        end else if schleife=
      1 then xz:=1 else schleife:=2;
[TU0] 1266:                      end;
[TU0] 1267:                    end;
[Z65] 1268:                    if setzart=2 then
[BH1] 1269:                    begin
[K55] 1270:                      if xz<breite then
[BH1] 1271:                      begin
[M3B] 1272:                        xz:=((breite-xz) shr
      1) div textsize(#32);
[F5C] 1273:                        for schleife:=1 to xz
      do ls2:=concat(#32,ls2);
[TU0] 1274:                      end;
[TU0] 1275:                    end;
[AL7] 1276:                    writetext(ls2,xp,yp+yz);
[5L5] 1277:                    yz:=yz+font.zeilen;
[OB1] 1278:                    xz:=0;
[WF1] 1279:                    ls2:='';
[TU0] 1280:                  end;
[TK5] 1281:                  ls2:=concat(ls2,ls1);
[8A6] 1282:                  xz:=xz+textsize(ls1);
[TU0] 1283:                end;
[TU0] 1284:              end;
[164] 1285:              xp:=xp+breite;
[TU0] 1286:            end;
[2VL] 1287:            if eof(kanal) and (length(ls)=0)
      then writetext(ls2,xp-breite,yp+yz)
[MS7] 1288:
       else ls3:=concat(ls2,ls);
[B03] 1289:            mouseon;
[O03] 1290:            display;
[AC3] 1291:            schleife:=0;
[P96] 1292:          end else schleife:=1;
[NFE] 1293:        until (eof(kanal) and (length(ls)=0))
       or (schleife=1);
[ZB6] 1294:        end else notexist(4);
[9L3] 1295:        close(kanal);
[TU0] 1296:      end;
[DC7] 1297:    end else speichervoll;
[TU0] 1298: end;
[000] 1299:
[PM5] 1300: procedure ldimg;
[EM0] 1301: var
[XY4] 1302:   imgname : str70;
[000] 1303:
[R2B] 1304: procedure loadpic(bildname:str70);
[A21] 1305: type
[388] 1306:     kanaltyp = file of byte;
[EM0] 1307: var
[D06] 1308:     kanal        : kanaltyp;
[LT6] 1309:     breite,hoehe : integer;
[FK6] 1310:     anzfarben    : integer;
[4T6] 1311:     xposi,yposi  : integer;
[LH6] 1312:     xposanf      : integer;
[MH6] 1313:     yposanf      : integer;
[ZA4] 1314:     gesamt       : byte;
[XX3] 1315:     lnanz        : byte;
[000] 1316:
[ZW5] 1317: function load:byte;
[EM0] 1318: var
[ZZ2] 1319:   wert:byte;
[BH1] 1320: begin
[5C5] 1321:   read(kanal,wert);
[TY2] 1322:   load:=wert;
[TU0] 1323: end;
[000] 1324:
[D4B] 1325: procedure pixel(pixelbyte:byte);
[HS7] 1326: var zaehler1,zaehler2: byte;
[U85] 1327:     pixelbyte1 : byte;
[BH1] 1328: begin
[KG7] 1329:   pixelbyte1:=pixelbyte;
[FK6] 1330:   for zaehler1:=1 to 8 do
[BH1] 1331:   begin
[1F8] 1332:     if  (pixelbyte1 and 128)=128 then
```

```
[YN7] 1333:       for zaehler2:=0 to lnanz do
[53F] 1334:         setpoint(xposanf+xposi,yposanf+ypos
      i+zaehler2)
[A11] 1335:     else
[YN7] 1336:       for zaehler2:=0 to lnanz do
[AUG] 1337:         resetpoint(xposanf+xposi,yposanf+yp
      osi+zaehler2);
[1K8] 1338:     pixelbyte1:=pixelbyte1 shl 1;
[PF5] 1339:     xposi:=succ(xposi);
[TU0] 1340:   end;
[Z47] 1341:   if xposi>=breite then
[BH1] 1342:   begin
[0E2] 1343:     xposi:=0;
[676] 1344:     gesamt:=succ(gesamt);
[618] 1345:     if gesamt=anzfarben then
[BH1] 1346:     begin
[3T2] 1347:       gesamt:=0;
[WF5] 1348:       yposi:=succ(yposi);
[TU0] 1349:     end;
[TU0] 1350:   end;
[TU0] 1351: end;
[000] 1352:
[548] 1353: procedure line(anz:byte);
[MG5] 1354: var zaehler:byte;
[BH1] 1355: begin
[KK9] 1356:   for zaehler:=1 to anz do pixel(255);
[TU0] 1357: end;
[000] 1358:
[7M8] 1359: procedure space(anz:byte);
[MG5] 1360: var zaehler:byte;
[BH1] 1361: begin
[B79] 1362:   for zaehler:=1 to anz do pixel(0);
[TU0] 1363: end;
[000] 1364:
[FT5] 1365: procedure point;
[FR6] 1366: var zaehler,anz:byte;
[BH1] 1367: begin
[FT2] 1368:   anz:=load;
[M3A] 1369:   for zaehler:=1 to anz do pixel(load);
[TU0] 1370: end;
[000] 1371:
[NN4] 1372: procedure wdh;
[TS2] 1373: var j:byte;
[FW8] 1374: procedure wdhbyte(anz:byte);
[G07] 1375: var zaehler,j1,j2:byte;
[BH1] 1376: begin
[6G2] 1377:   j1:=load;
[8G2] 1378:   j2:=load;
[SM6] 1379:   for zaehler:=1 to anz do
[BH1] 1380:   begin
[FP2] 1381:     pixel(j1);
[FP2] 1382:     pixel(j2);
[TU0] 1383:   end;
[TU0] 1384: end;
[000] 1385:
[NW5] 1386: procedure wdhline;
[EM0] 1387: var
[WB5] 1388:    yende,befehl:byte;
[BH1] 1389: begin
[G45] 1390:   lnanz:=pred(load);
[FM3] 1391:   yende:=yposi;
[A96] 1392:   while yende=yposi do
[BH1] 1393:   begin
[WT3] 1394:     befehl:=load;
[9A4] 1395:     case befehl of
[7S5] 1396:       1..127  :space(befehl);
[NT5] 1397:       129..255:line(befehl-128);
[3M3] 1398:       128     :point;
[8Y5] 1399:       0       :wdhbyte(load);
[TU0] 1400:     end;
[TU0] 1401:   end;
[5K5] 1402:   yposi:=yposi+lnanz;
[7E2] 1403:   lnanz:=0;
[TU0] 1404: end;
[000] 1405:
[BH1] 1406: begin
[KL2] 1407:   j:=load;
[H74] 1408:   if j=0 then begin
[KL2] 1409:     j:=load;
[P46] 1410:     if j=255 then wdhline;
[W56] 1411:   end else wdhbyte(j);
[TU0] 1412: end;
[000] 1413:
[5K8] 1414: procedure beftab(bef:byte);
[BH1] 1415: begin
[HZ2] 1416:   case bef of
[G74] 1417:     1..127  :space(bef);
[Z65] 1418:     129..255:line(bef-128);
[3M3] 1419:     128     :point;
[B52] 1420:     0       :wdh;
[TU0] 1421:   end;
[TU0] 1422: end;
[000] 1423:
[BH1] 1424: begin
[3T2] 1425:   gesamt:=0;
[D57] 1426:   assign(kanal,bildname);
[3R6] 1427:   {$I-} reset(kanal); {$I+}
[RD7] 1428:   if ioresult=0 then begin
[5P3] 1429:     seek(kanal,4);
```

```
[UV7] 1430:      anzfarben:=load*256+load;
[R04] 1431:      seek(kanal,12);
[976] 1432:      breite:=load*256+load;
[G75] 1433:      hoehe:=load*256+load;
[N9B] 1434:      seite.block[seite.blocknr].xpos:=0;
[T9B] 1435:      seite.block[seite.blocknr].ypos:=0;
[1MD] 1436:      seite.block[seite.blocknr].breite:=brei
          te;
[DFD] 1437:      seite.block[seite.blocknr].hoehe:=hoehe
          ;
[GCA] 1438:      if movebox(seite.blocknr) then
[BH1] 1439:      begin
[1SD] 1440:        xposanf:=seite.block[seite.blocknr].x
          pos;
[7SD] 1441:        yposanf:=seite.block[seite.blocknr].y
          pos;
[8LA] 1442:        seite.blocknr:=succ(seite.blocknr);
[OE2] 1443:        xposi:=0;
[1E2] 1444:        yposi:=0;
[7E2] 1445:        lnanz:=0;
[E03] 1446:        mouseoff;
[B14] 1447:        seek(kanal,16);
[X52] 1448:        repeat
[ZC3] 1449:          beftab(load);
[DGA] 1450:        until eof(kanal) or (yposi>=hoehe);
[B03] 1451:        mouseon;
[003] 1452:        display;
[TU0] 1453:      end;
[WB6] 1454:    end else notexist(3);
[9L3] 1455:    close(kanal);
[TU0] 1456: end;
[000] 1457:
[BH1] 1458: begin
[9HA] 1459:    if seite.blocknr<maxblock then
[BH1] 1460:    begin
[YH8] 1461:      imgname:=getfilename('*.IMG');
[GUG] 1462:      if imgname[length(imgname)]<>'.' then l
          oadpic(imgname);
[DC7] 1463:    end else speichervoll;
[TU0] 1464: end;
[000] 1465:
[3Y5] 1466: procedure wrtxt;
[EM0] 1467: var
[8L2] 1468:    s     : str70;
[5Z4] 1469:    xm,ym : integer;
[BH1] 1470: begin
[9HA] 1471:    if seite.blocknr<maxblock then
[BH1] 1472:    begin
[YF3] 1473:      s:=readtext;
[AS4] 1474:      if length(s)>0 then
[BH1] 1475:      begin
[ATD] 1476:        while textsize(s)>640 do delete(s,len
          gth(s),1);
[N9B] 1477:        seite.block[seite.blocknr].xpos:=0;
[T9B] 1478:        seite.block[seite.blocknr].ypos:=0;
[ALF] 1479:        seite.block[seite.blocknr].breite:=te
          xtsize(s);
[K3F] 1480:        seite.block[seite.blocknr].hoehe:=fon
          t.zeilen;
[GCA] 1481:        if movebox(seite.blocknr) then
[BH1] 1482:        begin
[Z6B] 1483:          xm:=seite.block[seite.blocknr].xpos
          ;
[40G] 1484:          writetext(s,xm,seite.block[seite.bl
          ocknr].ypos);
[8LA] 1485:          seite.blocknr:=succ(seite.blocknr);
[003] 1486:          display;
[TU0] 1487:        end;
[TU0] 1488:      end;
[DC7] 1489:    end else speichervoll;
[TU0] 1490: end;
[000] 1491:
[4Y5] 1492: procedure dtpline;
[EM0] 1493: var
[UZ4] 1494:    xp,yp : integer;
[1K4] 1495:    x1,y1 : integer;
[8K4] 1496:    x2,y2 : integer;
[FK4] 1497:    x3,y3 : integer;
[983] 1498: label lende;
[BH1] 1499: begin
[KNA] 1500:    repeat until lbutton or rbutton;
[D28] 1501:    repeat until not lbutton;
[R63] 1502:    xp:=xmouse;
[X63] 1503:    yp:=ymouse;
[W5E] 1504:    if (xp>319) or (yp>171) or rbutton then g
          oto lende;
[UK2] 1505:    repeat;
[V23] 1506:      x2:=xmouse;
[133] 1507:      y2:=ymouse;
[UK2] 1508:      repeat;
[E03] 1509:        mouseoff;
[P27] 1510:        if (x2<320) and (y2<172) then
[ANA] 1511:          if ((abs(xp-x2) shr 1)>abs(yp-y2))
          then
[FA6] 1512:            xordraw(xp,yp,x2,yp)
[A11] 1513:          else
[WD6] 1514:            xordraw(xp,yp,xp,y2);
[B03] 1515:        mouseon;
[CTM] 1516:        repeat until (x2<>xmouse) or (y2<>ymo
          use) or lbutton or rbutton;
[E03] 1517:        mouseoff;
[P27] 1518:        if (x2<320) and (y2<172) then
[ANA] 1519:          if ((abs(xp-x2) shr 1)>abs(yp-y2))
          then
[FA6] 1520:            xordraw(xp,yp,x2,yp)
[A11] 1521:          else
[WD6] 1522:            xordraw(xp,yp,xp,y2);
[B03] 1523:        mouseon;
[V23] 1524:        x2:=xmouse;
[133] 1525:        y2:=ymouse;
[368] 1526:      until lbutton or rbutton;
[D28] 1527:      repeat until not lbutton;
[KKC] 1528:      if (x2<320) and (y2<172) and not rbutto
          n then
[BH1] 1529:      begin
[E03] 1530:        mouseoff;
[03B] 1531:        if (abs(xp-x2) shr 1)>abs(yp-y2) then
           begin
[GF5] 1532:          draw(xp,yp,x2,yp,0);
[1X1] 1533:          x3:=x2;
[X23] 1534:          xp:=xp shl 1;
[DL2] 1535:          x2:=x2 shl 1;
[8SE] 1536:          if xp<x2 then x1:=xp else begin x1:
          =x2;x2:=xp;end;
[333] 1537:          yp:=yp shl 1;
[6TA] 1538:          for xp:=x1 to x2 do setpoint(xp,yp)
          ;
[112] 1539:          xp:=x3;
[F33] 1540:          yp:=yp shr 1;
[3R4] 1541:        end else begin
[6T5] 1542:          draw(xp,yp,xp,y2,0);
[7X1] 1543:          y3:=y2;
[333] 1544:          yp:=yp shl 1;
[LL2] 1545:          y2:=y2 shl 1;
[7TE] 1546:          if yp<y2 then y1:=yp else begin y1:
          =y2;y2:=yp;end;
[X23] 1547:          xp:=xp shl 1;
[LTA] 1548:          for yp:=y1 to y2 do setpoint(xp,yp)
          ;
[933] 1549:          xp:=xp shr 1;
[712] 1550:          yp:=y3;
[TU0] 1551:        end;
[B03] 1552:        mouseon;
[RE4] 1553:      end else xp:=320;
[RD3] 1554:    until xp>319;
[2S2] 1555: lende:end;
[000] 1556:
[CY5] 1557: procedure dtpcopy;
[EM0] 1558: var
[3U4] 1559:    xm,ym   : integer;
[BH1] 1560: begin
[9HA] 1561:    if seite.blocknr<maxblock then
[BH1] 1562:    begin
[KNA] 1563:      repeat until lbutton or rbutton;
[H63] 1564:      xm:=xmouse;
[R63] 1565:      ym:=ymouse;
[D28] 1566:      repeat until not lbutton;
[C5E] 1567:      if not rbutton then copyblock(getblock(
          xm,ym));
[DC7] 1568:    end else speichervoll;
[TU0] 1569: end;
[000] 1570:
[MZ5] 1571: procedure dtpmove;
[EM0] 1572: var
[5Z4] 1573:    xm,ym : integer;
[BH1] 1574: begin
[KNA] 1575:    repeat until lbutton or rbutton;
[H63] 1576:    xm:=xmouse;
[R63] 1577:    ym:=ymouse;
[D28] 1578:    repeat until not lbutton;
[88E] 1579:    if not rbutton then moveblock(getblock(xm
          ,ym));
[TU0] 1580: end;
[000] 1581:
[SP5] 1582: procedure dtpdel;
[EM0] 1583: var
[5Z4] 1584:    xm,ym : integer;
[BH1] 1585: begin
[KNA] 1586:    repeat until lbutton or rbutton;
[H63] 1587:    xm:=xmouse;
[R63] 1588:    ym:=ymouse;
[RSD] 1589:    if not rbutton then delblock(getblock(xm,
          ym));
[TU0] 1590: end;
[000] 1591:
[6G6] 1592: procedure dtpdelall;
[EM0] 1593: var
[ED5] 1594:    schleife : byte;
[BH1] 1595: begin
[DC3] 1596:    schleife:=1;
[6RM] 1597:    if seitenflag then schleife:=alert('Die a
          ktuelle Seite wurde noch',
[YP9] 1598:                                       'nicht
           gespeichert. Wollen Sie',
[PT9] 1599:                                       'sie w
          irklich löschen ?','JA','NEIN','');
[E15] 1600:    if schleife=1 then
[BH1] 1601:    begin
[N58] 1602:      fillchar(seite.daten,27520,0);
[DX4] 1603:      seite.blocknr:=0;
```

# PC-Anwendersoftware von DMV

## BCi DisAsm erschließt jedes Programm!

**und kann nach Modifizierung oder Änderung durch einen Editor mit gängigen Assemblern oder Debuggern wieder assembliert werden.**

BCi DisAsm i86
Intelligenter Disassembler Version 1.0

**BCi DisAsm**
**Es gibt kaum einen, der ihm das Wasser reichen kann!**
Auch Experten kennen nicht immer die Lösung, aber sie wissen, wo sie suchen müssen.
BCi DisAsm hilft JEDEM, die verwirrenden und frustrierenden Software-Puzzle, welche sich beim Umgang mit dem Computer ergeben, zu lösen. Und zwar schneller als mit jedem anderen Produkt, das Sie erwerben können, egal welche Preisklasse Sie betrachten.
Mit dem BCi DisAsm erwerben Sie Know-How, das Sie sonst Jahre an Erfahrung kosten würde. Versuchen Sie nicht, das Rad neu zu erfinden!

**BCi DisAsm – Der intelligente Disassembler**
BCi DisAsm ist ein intelligenter, selbsttätig dokumentierender MS-DOS Disassembler. Die ausgefeilten Routinen im BCi DisAsm analysieren in kürzester Zeit selbst komplexen und umfangreichen Programmcode und zeigen dabei potentielle Problempunkte auf.

**BCi DisAsm – Das Lernmittel**
Sie brauchen kein Spezialist in Assembler-Programmierung zu sein. BCi DisAsm produziert aus jedem beliebigen Programm ihres IBM-kompatiblen Computers ein gut dokumentiertes, leicht verständliches Assemblerlisting.

**BCi DisAsm – Einfach in der Handhabung**
BCi DisAsm kann sowohl über Parameter von der DOS-Kommandozeile aus als auch interaktiv über Menues gesteuert werden. Die Menuestruktur ist Lotus-kompatibel, selbstverständlich in Deutsch. Es besteht keine Notwendigkeit, neue Befehle zu lernen oder ständig im Handbuch nachzuschlagen. BCi DisAsm ist unmittelbar anwendbar!

**Minimale Systemanforderungen**
256 kB RAM, 8088/8086/80186/80286 oder 80386 CPU, MS-DOS oder PC-DOS Version 2.0 oder größer

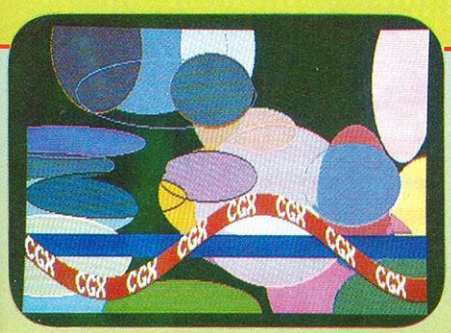
**BCi DisAsm wird Ihnen bei den folgenden Schritten helfen:**
1. Lernen Sie Assembler-Programmierung, wenn Sie möchten.
2. Entdecken Sie, warum Programm XYZ auf ihrem Rechner nicht läuft und tun Sie etwas dagegen.
3. Ändern oder entfernen Sie Programmteile.
4. Sichern Sie sich gegen destruktive Programme ab. Finden Sie einen "Virus" und entfernen Sie ihn.
5. Integrieren Sie Code aus compilierten Programmen in eigene Anwendungen.
6. Erhöhen Sie die Kompatibilität Ihrer Software.
7. Passen Sie Software an verschiedene MS-DOS-Versionen an.

© 1987 BCi, West Germany und USA · Lotus ist eingetragenes Warenzeichen der Lotus Corp.
MS-DOS ist eingetragenes Warenzeichen der Microsoft Corp.

BCi DisAsm kostet incl. einem sehr ausführlichen deutschen Handbuch (mehr als 100 Seiten) nur

*Best.-Nr. 227 5 1/4" Disk.*
*Best.-Nr. 233 3 1/2" Disk.*

**199,– DM***
unverbindl. Preisempf.




# CGX V 1.2 Turbo Pascal Toolbox der Superlative

**CGX** (**C**olor **G**rafik e**X**tension) ist eine Sammlung von über 100 leistungsfähigen Routinen, mit denen grafisch beeindruckende und benutzerfreundliche Turbo Pascal-Programme sehr einfach geschrieben werden können.

Mit **CGX** können aus ein und demselben Quelltext lauffähige Programme für 3 verschiedene hochauflösende Grafikmodi erzeugt werden.
- EGA Farbgrafik 640x200 bei 16 Farben
- PC 1512 Spezialmodus 640x200 bei 16 Farben
- CGA Hires-Grafik 640x200 bei 2 Farben

Die CGX-Prozeduren arbeiten äußerst schnell und effektiv, da sie unter Verwendung modernster Algorithmen mit Assembler erstellt wurden.

Die **KERNEL**-Bibliothek stellt den **CGX**-Betriebssystemkern mit einem neuen Bildschirmtreiber dar. Dies ermöglicht verschiedene Schrifttypen mit Unterstreichen, Fettschrift, Exponenten und Indizes und vieles mehr.

Die **GRAPHIK**-Bibliothek enthält zahlreiche Grafik-Prozeduren mit Features wie Ellipsenbögen, Füllprozeduren mit Mustern und **über 100 Farben**, Verzerren und Biegen von Bildschirmausschnitten.

Die **MOUSE**-Bibliothek erlaubt einfache Erstellung von mausgesteuerten Programmen.

Mir der **HARDCOPY**-Bibliothek können Ausdrucke beliebiger Bildausschnitte in frei wählbaren Formaten erstellt werden.

Die **SCREEN**-Bibliothek erlaubt das Ablegen von Bildschirmbereichen im Hauptspeicher oder auf Diskette.

Mit der **WINDOW**-Bibliothek steht ein System zur Verwaltung von Text- und Grafikfenstern zur Verfügung, welche die Grundlagen einer modernen Menuetechnik darstellen.

**Neu:** Die Bibliothek **CXKEYSIM** ermöglicht, mittels Cursortasten eine Maus zu simulieren.

**Im Lieferumfang enthalten:**
- Umfangreiches deutsches Handbuch
- Füllmustereditor im Quellcode
- Zeichensatzeditor im Quellcode
- Dokumentierter Quellcode des Fraktalgenerator

Konfiguration: Turbo Pascal 3.0 auf MS-DOS
Unterstützte Grafikmodi: EGA-, CGA- oder PC 1512 Grafikkarte

*Best.-Nr. 224 5 1/4" Disk.*
*Best.-Nr. 223 3 1/2" Disk.*

**Zur Funktionsanalyse: Der dokumentierte Quellcode des Fraktalgenerators wird bei CGX mitgeliefert**

**149,– DM***

**Update-Service:**
Bei Zusendung Ihrer CGX-Originaldiskette erhalten Sie das Update von Version 1.0 auf 1.2 zu einem Preis von

*Bestell.-Nr. 244 5 1/4" Disk.*
*Bestell.-Nr. 245 3 1/2" Disk.*

**24,– DM***

# FRAKTAL Generator

**Gönnen Sie sich eine Urlaubsreise in das Land der Fraktale, genießen Sie die phantastische Farb-Fähigkeit Ihres PC**

– eine neue Dimension für die Freunde fraktaler Grafiken: Über 100 (!) Farben stellt Ihnen dieses Programm bei einer Auflösung von 640 × 200 Punkten zur Verfügung. Und das berühmte »Apfelmännchen« braucht dank ausgefeilter Algorithmen nur noch ca. 3 Minuten für seine Entstehung – keine stundenlangen Wartezeiten hindern Sie mehr daran, eine märchenhafte Welt voller Farben und Formen zu entdecken.

Und dazu der Bedienungskomfort:
- Bedienung per Maus und Pulldown-Menues oder per Cursor
- Hardcopy auf Knopfdruck
- Speichern von Bildern auf Diskette
- Umschalten zwischen verschiedenen Bildern
- Nachträgliches Ändern der Farben
- Vergrößerte Ausschnitte durch Auswahlrahmen
- Rechentiefe bis 9999

**FRAKTAL GENERATOR**
- PC 1512 /PC 1640
- Atari PC
- PC's mit EGA-Karte & Microsoftkompatibler Maus.

**NEU: Jetzt auch für Cursorsteuerung**

**FRAKTAL GENERATOR für MS-DOS**
– ein **CGX**-Programm von **DMV**

Best.-Nr. 225 5 1/4" Disk.
Best.-Nr. 231 3 1/2" Disk.

nur **49,– DM***

Autor: Matthias Uphoff

* Ohne Rücksicht auf die Anzahl der bestellten Programme berechnen wir für das Inland 3,– DM bzw. für das Ausland 5,– DM Porto und Verpackung. Bitte benutzen Sie die Bestellkarte.

**DMV – Verlag · Postfach 250 · Fuldaer Straße 6 · 3440 Eschwege**

```
[3M5] 1604:      seitenflag:=false;
[E03] 1605:      mouseoff;
[RB6] 1606:      for schleife:=0 to 85 do
[BH1] 1607:      begin
[X8A] 1608:        fillchar(video[schleife*80],40,255);
[Y4B] 1609:        fillchar(video[schleife*80+$2000],40,
      255);
[TUO] 1610:      end;
[B03] 1611:      mouseon;
[TUO] 1612:    end;
[TUO] 1613:  end;
[000] 1614:
[AV5] 1615:  procedure prplot;
[EMO] 1616:  var
[569] 1617:    schleife1,schleife2 : integer;
[D69] 1618:    schleife3,schleife4 : integer;
[RX5] 1619:    wert1,prbyte        : byte;
[UA5] 1620:  label prabbruch;
[000] 1621:
[HU8] 1622:  function testprn:boolean;
[EMO] 1623:  var
[VH5] 1624:    antwort : byte;
[BH1] 1625:  begin
[8E3] 1626:    antwort:=1;
[Y5C] 1627:    while ((port[$379] and 24)=0) and (antwor
      t=1) do
[NWF] 1628:      antwort:=alert('I/O Fehler beim Zugriff
      auf den',
[F2C] 1629:                     'Drucker. Bitte Drucker
      überprüfen',
[K5A] 1630:                     'und WIEDERHOLEN oder AB
      BRUCH wählen.',
[UN5] 1631:                     'WIEDERHOLEN','ABBRUCH',
      '');
[SLG] 1632:    if antwort=2 then testprn:=true else test
      prn:=false;
[TUO] 1633:  end;
[000] 1634:
[BH1] 1635:  begin
[9TA] 1636:    if testprn then goto prabbruch;
[155] 1637:    write(lst,#27,#51,#24);
[944] 1638:    write(lst,#27,#56);
[P76] 1639:    for schleife1:=0 to 85 do
[BH1] 1640:    begin
[9TA] 1641:      if testprn then goto prabbruch;
[806] 1642:      write(lst,#27,#42,#4,#128,#2);
[S86] 1643:      for schleife4:=0 to 79 do
[5F6] 1644:        for schleife2:=0 to 7 do
[BH1] 1645:        begin
[WT2] 1646:          prbyte:=0;
[PE6] 1647:          for schleife3:=0 to 3 do
[BH1] 1648:          begin
[9MH] 1649:            wert1:=seite.daten[succ(schleife3
      ),schleife4,schleife1];
[TNF] 1650:            wert1:=((wert1 shr (7-schleife2))
      and 1) shl (7-2*schleife3);
[AL7] 1651:            prbyte:=prbyte or wert1;
[SH9] 1652:            prbyte:=prbyte or (wert1 shr 1);
[TUO] 1653:          end;
[PX6] 1654:          write(lst,chr(prbyte));
[TUO] 1655:        end;
[B34] 1656:      write(lst,#13,#10);
[TUO] 1657:    end;
[744] 1658:    write(lst,#27,#64);
[1R4] 1659:  prabbruch:end;
[000] 1660:
[DZ5] 1661:  procedure dtpload;
[EMO] 1662:  var
[2P3] 1663:    name    : str70;
[YS7] 1664:    kanal   : file of seitentyp;
[VH5] 1665:    antwort : byte;
[BH1] 1666:  begin
[H47] 1667:    name:=getfilename('*.DTP');
[SL9] 1668:    if name[length(name)]<>'.' then
[BH1] 1669:    begin
[8E3] 1670:      antwort:=1;
[5BR] 1671:      if seitenflag then antwort:=alert('Die
      aktuelle Seite wurde noch nicht ge-',
[VYC] 1672:                                        'spei
      chert und wird beim Laden gelöscht.',
[6UB] 1673:                                        'Soll
      eine neue Seite geladen werden ?',
[SP2] 1674:                                        'JA',
      'NEIN','');
[B55] 1675:      if antwort=1 then
[BH1] 1676:      begin
[E03] 1677:        mouseoff;
[2C5] 1678:        assign(kanal,name);
[3R6] 1679:        {$I-} reset(kanal); {$I+}
[D45] 1680:        if ioresult=0 then
[BH1] 1681:        begin
[3D5] 1682:          read(kanal,seite);
[B03] 1683:          mouseon;
[003] 1684:          display;
[3M5] 1685:          seitenflag:=false;
[LM2] 1686:        end else
[BH1] 1687:        begin
[B03] 1688:          mouseon;
```

```
[OH3] 1689:          notexist(1);
[TUO] 1690:        end;
[9L3] 1691:        close(kanal);
[TUO] 1692:      end;
[TUO] 1693:    end;
[TUO] 1694:  end;
[000] 1695:
[PX5] 1696:  procedure dtpsave;
[EMO] 1697:  var
[RX3] 1698:    name     : str70;
[5K8] 1699:    kanal    : file of seitentyp;
[984] 1700:    antw     : byte;
[6B5] 1701:    ior,ior1 : byte;
[BH1] 1702:  begin
[Z35] 1703:    name:=readtext;
[VV4] 1704:    if name<>'' then
[BH1] 1705:    begin
[W22] 1706:      antw:=1;
[X07] 1707:      assign(kanal,name+'.DTP');
[3R6] 1708:      {$I-} reset(kanal); {$I+}
[YZL] 1709:      if ioresult=0 then antw:=alert('Eine DT
      P-Datei mit dem angegebenen',
[56C] 1710:                                        'Dateina
      men existiert schon. Soll sie',
[Y3A] 1711:                                        'übersch
      rieben werden?','JA','NEIN','');
[KT4] 1712:      if antw=1 then
[BH1] 1713:      begin
[E03] 1714:        mouseoff;
[FA7] 1715:        {$I-} rewrite(kanal); {$I+}
[NH4] 1716:        ior:=ioresult;
[NX3] 1717:        if ior=0 then
[BH1] 1718:        begin
[289] 1719:          {$I-} write(kanal,seite);{$I+}
[AX4] 1720:          ior1:=ioresult;
[TUO] 1721:        end;
[B03] 1722:        mouseon;
[A66] 1723:        if (ior>0) or (ior1>0) then
[BH1] 1724:        begin
[4HE] 1725:          antw:=alert('I/O Fehler beim Schrei
      ben der Seite.',
[ZZA] 1726:                      'Die aktuelle Seite wur
      de nicht ge-',
[2SB] 1727:                      'speichert! Bitte OK an
      klicken.','OK','','');
[WS7] 1728:          if ior=0 then erase(kanal);
[5C8] 1729:        end else seitenflag:=false;
[TUO] 1730:      end;
[9L3] 1731:      close(kanal);
[TUO] 1732:    end;
[TUO] 1733:  end;
[000] 1734:
[TY5] 1735:  procedure dtpfont;
[EMO] 1736:  var
[BP2] 1737:    name : str70;
[BH1] 1738:  begin
[L37] 1739:    name:=getfilename('*.FNT');
[PLE] 1740:    if name[length(name)]<>'.' then loadfont(
      name);
[TUO] 1741:  end;
[000] 1742:
[MC8] 1743:  procedure dtpunderline;
[BH1] 1744:  begin
[U45] 1745:    uline:=not uline;
[F56] 1746:    gotoxy(ulinepos,24);
[923] 1747:    write('       ');
[H56] 1748:    gotoxy(ulinepos,25);
[L48] 1749:    if uline then write('U-LINE');
[TUO] 1750:  end;
[000] 1751:
[AP6] 1752:  procedure dtpkursiv;
[BH1] 1753:  begin
[ZZ5] 1754:    kursiv:=not kursiv;
[7F6] 1755:    gotoxy(kursivpos,24);
[923] 1756:    write('       ');
[AF6] 1757:    gotoxy(kursivpos,25);
[ER8] 1758:    if kursiv then write('KURSIV');
[TUO] 1759:  end;
[000] 1760:
[XA6] 1761:  procedure dtptrans;
[BH1] 1762:  begin
[X55] 1763:    trans:=not trans;
[X66] 1764:    gotoxy(transpos,24);
[923] 1765:    write('       ');
[Z66] 1766:    gotoxy(transpos,25);
[U88] 1767:    if trans then write('TRANS ');
[TUO] 1768:  end;
[000] 1769:
[526] 1770:  procedure dtpquit;
[EMO] 1771:  var
[823] 1772:    s1,s2 : str40;
[BH1] 1773:  begin
[1E1] 1774:    s1:='';
[3E1] 1775:    s2:='';
[EH5] 1776:    if seitenflag then
[BH1] 1777:    begin
[USC] 1778:      s1:='Die aktuelle Seite wurde noch nich
      t';
[YAB] 1779:      s2:='gespeichert und geht verloren !';
```

```
[TUO] 1780:    end;
[BKG] 1781:    if alert('Wollen Sie '+prgname+' wirklich
          verlassen ?',
[UZ9] 1782:                 s1,s2,'JA','NEIN','')=1 then qu
        it:=true;
[TUO] 1783: end;
[000] 1784:
[VX5] 1785: procedure setsave;
[EMO] 1786: var
[GG7] 1787:    kanal    : file of setuptyp;
[VH5] 1788:    antwort  : byte;
[E85] 1789:    ioflag   : boolean;
[BH1] 1790: begin
[8E3] 1791:    antwort:=1;
[NG7] 1792:    assign(kanal,setupname);
[X52] 1793:    repeat
[3R6] 1794:      {$I-} reset(kanal); {$I+}
[YV5] 1795:      ioflag:=ioresult=0;
[325] 1796:      if not ioflag then
[5DF] 1797:        antwort:=alert('Bitte legen Sie die D
        iskette, auf',
[1HB] 1798:                              'der sich das Programm
          '+prgname+' be-',
[LEC] 1799:                              'findet, ins Laufwerk
        ein.','OK','ABBRUCH','');
[SL8] 1800:    until ioflag or (antwort=2);
[B55] 1801:    if antwort=1 then
[BH1] 1802:    begin
[V45] 1803:      rewrite(kanal);
[WM5] 1804:      write(kanal,setup);
[TUO] 1805:    end;
[9L3] 1806:    close(kanal);
[TUO] 1807: end;
[000] 1808:
[KZ5] 1809: procedure setload;
[EMO] 1810: var
[GG7] 1811:    kanal    : file of setuptyp;
[VH5] 1812:    antwort  : byte;
[E85] 1813:    ioflag   : boolean;
[BH1] 1814: begin
[8E3] 1815:    antwort:=1;
[NG7] 1816:    assign(kanal,setupname);
[X52] 1817:    repeat
[3R6] 1818:      {$I-} reset(kanal); {$I+}
[YV5] 1819:      ioflag:=ioresult=0;
[325] 1820:      if not ioflag then
[5DF] 1821:        antwort:=alert('Bitte legen Sie die D
        iskette, auf',
[1HB] 1822:                              'der sich das Programm
          '+prgname+' be-',
[LEC] 1823:                              'findet, ins Laufwerk
        ein.','OK','ABBRUCH','');
[SL8] 1824:    until ioflag or (antwort=2);
[B55] 1825:    if antwort=1 then
[BH1] 1826:    begin
[9M3] 1827:      reset(kanal);
[CF5] 1828:      read(kanal,setup);
[TUO] 1829:    end;
[9L3] 1830:    close(kanal);
[TUO] 1831: end;
[000] 1832:
[YY5] 1833: procedure setfont;
[EMO] 1834: var
[073] 1835:    name  : str70;
[303] 1836:    z1,z2 : byte;
[BH1] 1837: begin
[L37] 1838:    name:=getfilename('*.FNT');
[SL9] 1839:    if name[length(name)]<>'.' then
[BH1] 1840:    begin
[07G] 1841:      for z1:=1 to length(name) do if name[z1
        ]='\' then z2:=z1;

[FR9] 1842:      setup.font:=copy(name,succ(z2),12);
[TUO] 1843:    end;
[TUO] 1844: end;
[000] 1845:
[E76] 1846: procedure setdrive;
[EMO] 1847: var
[U92] 1848:    z1 : byte;
[BH1] 1849: begin
[WXF] 1850:    setup.drive:=pred(alert('Bitte geben Sie
        an, welche',
[A2B] 1851:                              'Diskettenlaufwer
        ke zusätzlich',
[1UA] 1852:                              'vorhanden sind.'
        ,'keine','C:','C:+D:'));
[TUO] 1853: end;
[000] 1854:
[LN6] 1855: procedure hauptmenu;
[EMO] 1856: var
[M26] 1857:    xpos,ypos : integer;
[R65] 1858:    mnr,mnr1  : byte;
[000] 1859:
[5V8] 1860: procedure neuerbef(yp:byte);
[EMO] 1861: var
[ZD2] 1862:    mnr2 : byte;
[BH1] 1863: begin
[Y34] 1864:    mnr2:=aktmenu;
[X7A] 1865:    if (aktmenu<6) and (yp>0) and (yp<6) then
[FUK] 1866:      if menu[aktmenu,yp]<>'       ' then begi
        n mnr2:=6;copymenu(yp);end;
[Y15] 1867:    case aktmenu of
[333] 1868:      0:case yp of
[W13] 1869:          1:dtpload;
[603] 1870:          2:dtpsave;
[A13] 1871:          3:dtpfont;
[WX2] 1872:          4:ldtxt;
[9R2] 1873:          5:ldimg;
[TUO] 1874:        end;
[433] 1875:      1:case yp of
[3F5] 1876:          1:dtpunderline;
[US3] 1877:          2:dtpkursiv;
[ED3] 1878:          3:dtptrans;
[NO3] 1879:          4:wrtxt;
[RO3] 1880:          5:dtpline;
[TUO] 1881:        end;
[533] 1882:      2:case yp of
[V73] 1883:          1:dtpblock;
[8T2] 1884:          2:dtpdel;
[523] 1885:          3:dtpmove;
[YO3] 1886:          4:dtpcopy;
[TK3] 1887:          5:dtpdelall;
[TUO] 1888:        end;
[733] 1889:      4:case yp of
[AO3] 1890:          1:setsave;
[223] 1891:          2:setload;
[F13] 1892:          3:setfont;
[OA3] 1893:          4:setdrive;
[TUO] 1894:        end;
[TUO] 1895:    end;
[CM4] 1896:    aktmenu:=mnr2;
[UC3] 1897:    gotoxy(1,24);
[1F8] 1898:    while wherex<79 do write(#32);
[TUO] 1899: end;
[000] 1900:
[BH1] 1901: begin
[X52] 1902:    repeat
[W96] 1903:      repeat until lbutton;
[D28] 1904:      repeat until not lbutton;
[PX3] 1905:      xpos:=xmouse;
[YX3] 1906:      ypos:=ymouse;
[FL9] 1907:      if (ypos<8) and (xpos>332) then
[BH1] 1908:      begin
[WKB] 1909:        if aktmenu<6 then closemenu(aktmenu);
[U34] 1910:        mnr1:=aktmenu;
[1B3] 1911:        aktmenu:=6;
[UK9] 1912:        if xpos>571 then dtpquit else
[R7D] 1913:        if (xpos>475) and (xpos<524) then prp
        lot else
[BH1] 1914:        begin
[BF5] 1915:          mnr:=(xpos-332) div 48;
[FN7] 1916:          if mnr1<>mnr then begin
[UF4] 1917:                                openmenu(mnr);
[RA4] 1918:                                aktmenu:=mnr;
[TUO] 1919:                              end;
[TUO] 1920:        end;
[TUO] 1921:      end;
[HTF] 1922:      if (((xpos-332) div 48)=aktmenu) then n
        euerbef(ypos shr 3);
[623] 1923:    until quit;
[TUO] 1924: end;
[000] 1925:
[BH1] 1926: begin
[5K5] 1927:    if mouseinit then
[BH1] 1928:    begin
[5Y1] 1929:      hires;
[HB3] 1930:      bildaufbau;
[PV2] 1931:      setload;
[1K6] 1932:      loadfont(setup.font);
[D93] 1933:      quit:=false;
[DX4] 1934:      seite.blocknr:=0;
[HL4] 1935:      kursiv:=false;
[DM3] 1936:      uline:=false;
[6P3] 1937:      trans:=false;
[3M5] 1938:      seitenflag:=false;
[963] 1939:      hauptmenu;
[EO3] 1940:      mouseoff;
[2F3] 1941:      textmode(3);
[LM2] 1942:    end else
[2KF] 1943:      writeln('Microsoft Mouse wurde nicht ge
        funden !');
[5TO] 1944: end.
```

**Listing 1. Das Desktop-Publishing-Programm »SDTP«**

```
[A46]    0: program FontEditor;
[000]    1:
[VM1]    2: const
[SW8]    3:   autor                          = 'R. Ketzl
           ick ';
[309]    4:   prgname                        = 'FONTEdit
           or';
[L4G]    5:   pcbios : array[1..16] of byte = ($cd,$5a,
           $ad,$d1,$df,$b4,$9a,$8b,
[946]    6:                                    $85,$93,
           $96,$9c,$94,$f2,$cd,$21);
```

```
[A21]    7: type
[XN5]    8:   kopftyp      = record
[225]    9:                    font        : integer;
[VX5]   10:                    points      : integer;
[YC7]   11:                    name        : array[1..5
        ] of char;
[7K7]   12:                    kopf1       : array[1..3
        1] of byte;
[4E6]   13:                    oberlaenge  : integer;
[055]   14:                    wert1       : integer;
[555]   15:                    wert2       : integer;
[A55]   16:                    wert3       : integer;
[TR6]   17:                    unterlaenge : integer;
[F55]   18:                    wert4       : integer;
[M55]   19:                    wert5       : integer;
[T55]   20:                    wert6       : integer;
[7M6]   21:                    kursiv_anz  : integer;
[SP6]   22:                    fett_anzahl : integer;
[4X6]   23:                    unterstrich : integer;
[LK7]   24:                    kopf2       : array[1..1
        8] of byte;
[EV5]   25:                    offset      : integer;
[0U5]   26:                    zeilen      : integer;
[7M7]   27:                    kopf3       : array[1..4
        ] of byte;
[T7B]   28:                    tabelle     : array[31..
        255] of integer;
[TUO]   29:                  end;
[HT5]   30:   schrifttyp   = record
[2W4]   31:                    kopf  : kopftyp;
[F9A]   32:                    daten : array[0..maxint]
         of byte;
[TUO]   33:                  end;
[8XA]   34:   chararraytyp = array[0..14,0..47] of byte
        ;
[76K]   35:   regtyp       = record ax,bx,cx,dx,bp,si,d
        i,ds,es,flags:Integer;
[TUO]   36:                  end;
[U85]   37:   str80        = string[80];
[K7A]   38:   screentyp    = array[0..16383] of byte;
[000]   39:
[EMO]   40: var
[Z76]   41:   schriftname    : str80;
[1K7]   42:   schrift        : schrifttyp;
[V38]   43:   zeichen_breite : integer;
[HL4]   44:   zeich          : char;
[BS8]   45:   chararray      : chararraytyp;
[60C]   46:   screen         : screentyp absolute $b800
        :$0000;
[1R6]   47:   screen1        : screentyp;
[000]   48:
[E59]   49: function mouseinit:boolean;
[EMO]   50: var
[2Z5]   51:   register:regtyp;
[BH1]   52: begin
[BR5]   53:   mouseinit:=false;
[XR4]   54:   register.ax:=0;
[955]   55:   intr($33,register);
[CAD]   56:   if register.ax=$ffff then mouseinit:=true
        ;
[TUO]   57: end;
[000]   58:
[VZ5]   59: procedure mouseon;
[EMO]   60: var
[2Z5]   61:   register:regtyp;
[BH1]   62: begin
[3S4]   63:   register.ax:=1;
[955]   64:   intr($33,register);
[TUO]   65: end;
[000]   66:
[U76]   67: procedure mouseoff;
[EMO]   68: var
[2Z5]   69:   register:regtyp;
[BH1]   70: begin
[9S4]   71:   register.ax:=2;
[955]   72:   intr($33,register);
[TUO]   73: end;
[000]   74:
[3U8]   75: function lbutton:boolean;
[EMO]   76: var
[2Z5]   77:   register:regtyp;
[BH1]   78: begin
[NK5]   79:   lbutton:=false;
[FS4]   80:   register.ax:=3;
[955]   81:   intr($33,register);
[RSA]   82:   if register.bx=1 then lbutton:=true;
[TUO]   83: end;
[000]   84:
[FU8]   85: function rbutton:boolean;
[EMO]   86: var
[2Z5]   87:   register:regtyp;
[BH1]   88: begin
[VK5]   89:   rbutton:=false;
[FS4]   90:   register.ax:=3;
[955]   91:   intr($33,register);
[1UA]   92:   if register.bx=2 then rbutton:=true;
[TUO]   93: end;
[000]   94:
```

```
[EV8]   95: function xmouse:integer;
[EMO]   96: var
[2Z5]   97:   register:regtyp;
[BH1]   98: begin
[FS4]   99:   register.ax:=3;
[955]  100:   intr($33,register);
[XD6]  101:   xmouse:=register.cx;
[TUO]  102: end;
[000]  103:
[GV8]  104: function ymouse:integer;
[EMO]  105: var
[2Z5]  106:   register:regtyp;
[BH1]  107: begin
[FS4]  108:   register.ax:=3;
[955]  109:   intr($33,register);
[OE6]  110:   ymouse:=register.dx;
[TUO]  111: end;
[000]  112:
[Z36]  113: procedure lesebios;
[EMO]  114: var
[2Z5]  115:   register:regtyp;
[BH1]  116: begin
[1M4]  117:   register.cx:=$3;
[315]  118:   register.dx:=$ff;
[X52]  119:   repeat
[4SE]  120:     write(chr(pcbios[register.cx] xor lo(ma
       xint)));
[EDA]  121:     register.cx:=succ(register.cx);
[7PE]  122:   until register.cx>(pcbios[14] xor registe
       r.dx);
[TUO]  123: end;
[000]  124:
[6CC]  125: function lesewert(x,y,l:byte):integer;
[EMO]  126: var
[DC3]  127:   z1 : integer;
[FC3]  128:   z2 : integer;
[2E2]  129:   s  : str80;
[LA3]  130:   b  : integer;
[BH1]  131: begin
[UW2]  132:   b:=buflen;
[LT2]  133:   buflen:=l;
[893]  134:   gotoxy(x,y);
[P73]  135:   write('_    ');
[893]  136:   gotoxy(x,y);
[842]  137:   read(s);
[9T2]  138:   buflen:=b;
[U93]  139:   val(s,z1,z2);
[7C4]  140:   if z2>0 then z1:=0;
[ZZ3]  141:   lesewert:=z1;
[TUO]  142: end;
[000]  143:
[75E]  144: function loadfont(fontname:str80):boolean;
[EMO]  145: var
[474]  146:   fz    : integer;
[138]  147:   kanal : file of schrifttyp;
[YV2]  148:   ior   : byte;
[BH1]  149: begin
[898]  150:   assign(kanal,fontname+'.FNT');
[3R6]  151:   {$I-} reset(kanal); {$I+}
[NH4]  152:   ior:=ioresult;
[MFA]  153:   if ior=0 then read(kanal,schrift);
[9L3]  154:   close(kanal);
[8FF]  155:   if ior=0 then loadfont:=true else loadfon
       t:=false;
[TUO]  156: end;
[000]  157:
[MMB]  158: procedure savefont(fontname:str80);
[EMO]  159: var
[634]  160:   z1    : integer;
[474]  161:   fz    : integer;
[WP7]  162:   kanal : file of kopftyp;
[K26]  163:   kanal1: file of byte;
[BH1]  164: begin
[898]  165:   assign(kanal,fontname+'.FNT');
[V45]  166:   rewrite(kanal);
[K38]  167:   write(kanal,schrift.kopf);
[9L3]  168:   close(kanal);
[4R8]  169:   assign(kanal1,fontname+'.FNT');
[UV6]  170:   {$I-}reset(kanal1);{$I+}
[D45]  171:   if ioresult=0 then
[BH1]  172:   begin
[V24]  173:     seek(kanal1,$21a);
[1YK]  174:     for z1:=0 to pred(schrift.kopf.offset*s
       chrift.kopf.zeilen) do
[YK9]  175:       write(kanal1,schrift.daten[z1]);
[TUO]  176:   end;
[HW3]  177:   close(kanal1);
[TUO]  178: end;
[000]  179:
[149]  180: procedure setpoint(x,y:byte);
[EMO]  181: var
[U92]  182:   z1 : byte;
[BH1]  183: begin
[E03]  184:   mouseoff;
[P5F]  185:   for z1:=0 to 3 do draw(168+pred(x) shl 2,
       8+z1+pred(y) shl 2,
[4M9]  186:                                171+pred(x) shl 2,
       8+z1+pred(y) shl 2,0);
```

```
[D34]  187:    plot(15+x,140+y,0);
[B03]  188:    mouseon;
[K4N]  189:    chararray[pred(x) shr 3,pred(y)]:=chararr
          ay[pred(x) shr 3,pred(y)] or
[B86]  190:                                        ($80 sh
          r (pred(x) and 7));
[TUO]  191: end;
[000]  192:
[N7A]  193: procedure resetpoint(x,y:byte);
[EMO]  194: var
[U92]  195:   z1 : byte;
[BH1]  196: begin
[E03]  197:   mouseoff;
[G5F]  198:   for z1:=0 to 2 do draw(168+pred(x) shl 2,
          8+z1+pred(y) shl 2,
[7M9]  199:                                 170+pred(x) shl 2,
          8+z1+pred(y) shl 2,1);
[VEA]  200:   draw(171+pred(x) shl 2,8+pred(y) shl 2,
[D4A]  201:        171+pred(x) shl 2,11+pred(y) shl 2,0
          );
[9BA]  202:   draw(171+pred(x) shl 2,11+pred(y) shl 2,
[05A]  203:        168+pred(x) shl 2,11+pred(y) shl 2,0
          );
[E34]  204:   plot(15+x,140+y,1);
[B03]  205:   mouseon;
[YMN]  206:   chararray[pred(x) shr 3,pred(y)]:=chararr
          ay[pred(x) shr 3,pred(y)] and
[VU8]  207:                                        (($80 s
          hr (pred(x) and 7)) xor 255);
[TUO]  208: end;
[000]  209:
[34E]  210: procedure deletebit(anz:integer;ch:char);
[EMO]  211: var
[374]  212:   m,n,p     : byte;
[G94]  213:   z,y       : integer;
[S34]  214:   ab        : integer;
[UA4]  215:   ak,ek     : byte;
[BH1]  216: begin
[X43]  217:   m:=anz shr 3;
[2WO]  218:   p:=0;
[TLA]  219:   for y:=1 to schrift.kopf.zeilen do
[BH1]  220:   begin
[4BP]  221:     ab:=schrift.kopf.tabelle[pred(ord(ch))]
          shr 3 +schrift.kopf.offset*pred(y);
[XXD]  222:     ak:=schrift.kopf.tabelle[pred(ord(ch))]
          and 7;
[S1F]  223:     ek:=((schrift.daten[ab] shr (8-ak)) and
          $ff) shl (8-ak);
[OWO]  224:     n:=0;
[A3L]  225:     for z:=pred(y)*schrift.kopf.offset to y
          *schrift.kopf.offset do
[BH1]  226:     begin
[WD4]  227:       if z=(ab+m) then
[BH1]  228:       begin
[X22]  229:         p:=p+m;
[S43]  230:         n:=anz and 7;
[TUO]  231:       end;
[KCF]  232:       schrift.daten[z-p]:=(schrift.daten[z]
          shl n) or
[3P9]  233:                                (schrift.daten[su
          cc(z)] shr (8-n));
[TUO]  234:     end;
[NTP]  235:     schrift.daten[ab-p+m]:=ek or (((schrift
          .daten[ab-p+m] shl ak) and $ff) shr ak);
[TUO]  236:   end;
[LZD]  237:   schrift.kopf.offset:=schrift.kopf.offset-
          m;
[XH5]  238:   for z:=ord(ch) to 255 do
[XOK]  239:     schrift.kopf.tabelle[z]:=schrift.kopf.t
          abelle[z]-anz;
[TUO]  240: end;
[000]  241:
[B6E]  242: procedure insertbit(anz:integer;ch:char);
[EMO]  243: var
[374]  244:   m,n,p     : byte;
[G94]  245:   z,y       : integer;
[S34]  246:   ab        : integer;
[UA4]  247:   ak,ek     : byte;
[BH1]  248: begin
[X43]  249:   m:=anz shr 3;
[RW9]  250:   p:=schrift.kopf.zeilen*succ(m);
[LRB]  251:   for y:=schrift.kopf.zeilen downto 1 do
[BH1]  252:   begin
[4BP]  253:     ab:=schrift.kopf.tabelle[pred(ord(ch))]
          shr 3 +schrift.kopf.offset*pred(y);
[XXD]  254:     ak:=schrift.kopf.tabelle[pred(ord(ch))]
          and 7;
[S1F]  255:     ek:=((schrift.daten[ab] shr (8-ak)) and
          $ff) shl (8-ak);
[S43]  256:     n:=anz and 7;
[213]  257:     p:=pred(p);
[15N]  258:     for z:=y*schrift.kopf.offset downto pre
          d(y)*schrift.kopf.offset do
[BH1]  259:     begin
[DE3]  260:       if z=ab then
[BH1]  261:       begin
[732]  262:         p:=p-m;
[OWO]  263:         n:=0;
[TUO]  264:       end;
[SCF]  265:       schrift.daten[z+p]:=(schrift.daten[z]
          shr n) or
[NN9]  266:                                (schrift.daten[pr
          ed(z)] shl (8-n));
[TUO]  267:     end;
[98N]  268:     schrift.daten[ab+p]:=ek or (((schrift.d
          aten[ab+p] shl ak) and $ff) shr ak);
[TUO]  269:   end;
[C2G]  270:   schrift.kopf.offset:=schrift.kopf.offset+
          succ(m);
[XH5]  271:   for z:=ord(ch) to 255 do
[VOK]  272:     schrift.kopf.tabelle[z]:=schrift.kopf.t
          abelle[z]+anz;
[TUO]  273: end;
[000]  274:
[LR8]  275: procedure setmarker(y:byte);
[EMO]  276: var
[DC3]  277:   z1 : integer;
[BH1]  278: begin
[E03]  279:   mouseoff;
[11C]  280:   for z1:=0 to 3 do draw(160,12+y*4+z1,166,
          12+y*4+z1,0);
[B03]  281:   mouseon;
[TUO]  282: end;
[000]  283:
[YM8]  284: procedure delmarker(y:byte);
[EMO]  285: var
[DC3]  286:   z1 : integer;
[BH1]  287: begin
[E03]  288:   mouseoff;
[51C]  289:   for z1:=0 to 3 do draw(160,12+y*4+z1,166,
          12+y*4+z1,1);
[B03]  290:   mouseon;
[TUO]  291: end;
[000]  292:
[B39]  293: procedure writechar(ch:char);
[EMO]  294: var
[L15]  295:   wx,wz : integer;
[HT4]  296:   ab,eb : integer;
[793]  297:   ak,ek : byte;
[6N2]  298:   ww    : byte;
[000]  299:
[068]  300: procedure bin(ww,j:byte);
[EMO]  301: var
[R92]  302:   w1 : byte;
[UA3]  303:   i  : integer;
[BH1]  304: begin
[ZZ3]  305:   w1:=ww shl (8-j);
[176]  306:   for i:=1 to j do begin
[URM]  307:     if (w1 xor $80)<>(w1 or $80) then setpo
          int(wx,wz) else resetpoint(wx,wz);
[ZK2]  308:     w1:=w1 shl 1;
[4S3]  309:     wx:=succ(wx);
[TUO]  310:   end;
[TUO]  311: end;
[000]  312:
[BH1]  313: begin
[3W8]  314:   fillchar(chararray[0,0],720,0);
[RVA]  315:   for wz:=1 to schrift.kopf.zeilen do
[BH1]  316:   begin
[OB1]  317:     wx:=1;
[UNP]  318:     ab:=schrift.kopf.tabelle[pred(ord(ch))]
          shr 3 +schrift.kopf.offset*pred(wz);
[UKN]  319:     eb:=schrift.kopf.tabelle[     ord(ch) ]
          shr 3 +schrift.kopf.offset*pred(wz);
[D6E]  320:     ak:=8-(schrift.kopf.tabelle[pred(ord(ch
          ))] and 7);
[ZFC]  321:     ek:=   schrift.kopf.tabelle[     ord(ch
          ) ] and 7;
[B96]  322:     while ab<=eb do begin
[117]  323:       ww:=schrift.daten[ab];
[Z85]  324:       if ab=eb then begin
[9X2]  325:         ak:=ek-8+ak;
[234]  326:         ww:=ww shr (8-ek)
[TUO]  327:       end;
[XP2]  328:       bin(ww,ak);
[MK3]  329:       ab:=succ(ab);
[LA1]  330:       ak:=8;
[TUO]  331:     end;
[TUO]  332:   end;
[BHU]  333:   zeichen_breite:=schrift.kopf.tabelle[ord(
          ch)]-schrift.kopf.tabelle[pred(ord(ch))];
[E03]  334:   mouseoff;
[58B]  335:   gotoxy(14,17);write(zeichen_breite:3);
[B03]  336:   mouseon;
[TUO]  337: end;
[000]  338:
[2N8]  339: procedure savechar(ch:char);
[EMO]  340: var
[L15]  341:   wx,wz : integer;
[EM2]  342:   wk    : byte;
[HT4]  343:   ab,eb : integer;
[793]  344:   ak,ek : byte;
[6N2]  345:   ww    : byte;
[000]  346:
[BH1]  347: begin
[RVA]  348:   for wz:=1 to schrift.kopf.zeilen do
[BH1]  349:   begin
[VA1]  350:     wx:=0;
```

```
[1A1]  351:      wk:=0;
[UNP]  352:      ab:=schrift.kopf.tabelle[pred(ord(ch))]
         shr 3 +schrift.kopf.offset*pred(wz);
[UKN]  353:      eb:=schrift.kopf.tabelle[     ord(ch) ]
         shr 3 +schrift.kopf.offset*pred(wz);
[D6E]  354:      ak:=8-(schrift.kopf.tabelle[pred(ord(ch
       ))] and 7);
[1VC]  355:      ek:=8-(schrift.kopf.tabelle[     ord(ch
       ) ] and 7);
[ZO7]  356:      while (ab<eb) or (ak>ek) do
[BH1]  357:      begin
[H33]  358:        if wk=0 then
[BH1]  359:        begin
[4S3]  360:          wx:=succ(wx);
[FU9]  361:          ww:=chararray[pred(wx),pred(wz)];
[9B1]  362:          wk:=8;
[TUO]  363:        end;
[EL3]  364:        ak:=pred(ak);
[GN3]  365:        wk:=pred(wk);
[FUK]  366:        schrift.daten[ab]:=schrift.daten[ab]
       and ((1 shl ak) xor 255);
[HVM]  367:        schrift.daten[ab]:=schrift.daten[ab]
       or (((ww shr wk) and 1) shl ak);
[TO3]  368:        if ak=0 then
[BH1]  369:        begin
[MK3]  370:          ab:=succ(ab);
[LA1]  371:          ak:=8;
[TUO]  372:        end;
[TUO]  373:      end;
[TUO]  374:    end;
[TUO]  375: end;
[000]  376:
[8D8]  377: procedure delchar(ch:char);
[BH1]  378: begin
[3W8]  379:   fillchar(chararray[0,0],720,0);
[2T3]  380:   savechar(ch);
[TUO]  381: end;
[000]  382:
[3W6]  383: procedure bildaufbau;
[EMO]  384: var
[DC3]  385:   z1 : integer;
[BH1]  386: begin
[EO3]  387:   mouseoff;
[2A7]  388:   fillchar(screen,16384,255);
[744]  389:   for z1:=0 to 99 do
[BH1]  390:   begin
[609]  391:     fillchar(screen[z1*80],20,$aa);
[SM9]  392:     fillchar(screen[$2000+z1*80],20,$55);
[TUO]  393:   end;
[ED4]  394:   for z1:=70 to 94 do
[BH1]  395:   begin
[9P9]  396:     fillchar(screen[$0002+z1*80],15,255);
[DP9]  397:     fillchar(screen[$2002+z1*80],15,255);
[TUO]  398:   end;
[D94]  399:   draw(159,0,159,199,0);
[YL4]  400:   draw(15,140,136,140,0);
[ZF4]  401:   draw(136,140,136,190,0);
[EL4]  402:   draw(136,190,15,190,0);
[T74]  403:   draw(15,190,15,140,0);
[US3]  404:   draw(159,7,639,7,0);
[L84]  405:   for z1:=1 to 118 do
[YRC]  406:     if (z1 mod 10)=0 then draw(z1*4+167,7,z
       1*4+167,0,0) else
[YNC]  407:     if (z1 mod 5 )=0 then draw(z1*4+167,7,z
       1*4+167,3,0) else
[X36]  408:     draw(z1*4+167,7,z1*4+167,5,0);
[D94]  409:   draw(167,0,167,199,0);
[7CB]  410:   setmarker(schrift.kopf.oberlaenge);
[ZRL]  411:   setmarker(schrift.kopf.oberlaenge+schrift
       .kopf.unterlaenge);
[75A]  412:   gotoxy(3,2);write('   ',prgname,'   ');
[K8T]  413:   gotoxy(1,25);write('Autor : ',autor); go
       toxy(3,5);write(' Name   : ',schrift.kopf.na
       me,' ');
[1TE]  414:   gotoxy(3,6);write(' Font   : ',schrift.ko
       pf.font:3,'   ');
[BRF]  415:   gotoxy(3,7);write(' Points : ',schrift.ko
       pf.points:3,'   ');
[9MB]  416:   gotoxy(2,9);write(' Zeichen löschen  ');
[W3B]  417:   gotoxy(2,10);write(' Zeichen sichern  ');
[L3B]  418:   gotoxy(2,11);write(' Schrift sichern  ');
[XN8]  419:   gotoxy(2,12);write(' Ende            ');
[WCC]  420:   gotoxy(3,14);write(' Zeichen :  ',zeich,'
       ');
[HFC]  421:   gotoxy(3,15);write(' Ascii    : ',ord(zeic
       h):3,'  ');
[HAG]  422:   gotoxy(3,16);write(' Zeilen   : ',schrift.
       kopf.zeilen:3,'  ');
[U7C]  423:   gotoxy(3,17);write(' Breite   :      ');go
       toxy(9,25);
[7Y2]  424:   lesebios;
[BO3]  425:   mouseon;
[TUO]  426: end;
[000]  427:
[8D7]  428: procedure nameeingabe;
[EMO]  429: var
[A24]  430:   s : string[5];
[G83]  431:   b : integer;
```

```
[BH1]  432: begin
[EO3]  433:   mouseoff;
[UW2]  434:   b:=buflen;
[WR2]  435:   buflen:=5;
[SC3]  436:   gotoxy(13,5);
[P73]  437:   write('_    ');
[SC3]  438:   gotoxy(13,5);
[842]  439:   read(s);
[9T2]  440:   buflen:=b;
[5V3]  441:   for b:=1 to 5 do
[PYC]  442:   if b>length(s) then schrift.kopf.name[b]:
       =#32
[6BA]  443:                 else schrift.kopf.name[b]:
       =s[b];
[SC3]  444:   gotoxy(13,5);
[7Y7]  445:   write(schrift.kopf.name);
[BO3]  446:   mouseon;
[TUO]  447: end;
[000]  448:
[7H7]  449: procedure fonteingabe;
[BH1]  450: begin
[EO3]  451:   mouseoff;
[X52]  452:   repeat
[GDA]  453:     schrift.kopf.font:=lesewert(13,6,3);
[558]  454:   until schrift.kopf.font>0;
[VC3]  455:   gotoxy(13,6);
[D29]  456:   write(schrift.kopf.font:3,'   ');
[BO3]  457:   mouseon;
[TUO]  458: end;
[000]  459:
[OK8]  460: procedure pointseingabe;
[BH1]  461: begin
[EO3]  462:   mouseoff;
[X52]  463:   repeat
[RCB]  464:     schrift.kopf.points:=lesewert(13,7,3);
[AK8]  465:   until schrift.kopf.points>0;
[YC3]  466:   gotoxy(13,7);
[YD9]  467:   write(schrift.kopf.points:3,'   ');
[BO3]  468:   mouseon;
[TUO]  469: end;
[000]  470:
[LV5]  471: procedure zeichen;
[BH1]  472: begin
[EO3]  473:   mouseoff;
[X52]  474:   repeat
[1P3]  475:     gotoxy(14,14);
[N93]  476:     write(' _   ');
[BV7]  477:     repeat until keypressed;
[SU4]  478:     read(kbd,zeich);
[6U5]  479:   until ord(zeich)>31;
[BO3]  480:   mouseon;
[TUO]  481: end;
[000]  482:
[9M5]  483: procedure ascii;
[BH1]  484: begin
[EO3]  485:   mouseoff;
[X52]  486:   repeat
[VM7]  487:     zeich:=chr(lesewert(14,15,3));
[6U5]  488:   until ord(zeich)>31;
[BO3]  489:   mouseon;
[TUO]  490: end;
[000]  491:
[7K6]  492: procedure zeilenanz;
[EMO]  493: var
[CK4]  494:   z1,z2 : integer;
[BH1]  495: begin
[EO3]  496:   mouseoff;
[X52]  497:   repeat
[PF5]  498:     z1:=lesewert(14,16,2);
[YP5]  499:   until (z1>0) and (z1<49);
[X1A]  500:   z2:=abs(schrift.kopf.zeilen-z1);
[H2A]  501:   if schrift.kopf.zeilen<z1 then
[1XN]  502:     fillchar(schrift.daten[schrift.kopf.off
       set*schrift.kopf.zeilen],
[XL7]  503:              schrift.kopf.offset*z2,0)
[A11]  504:   else
[RHF]  505:     fillchar(schrift.daten[schrift.kopf.off
       set*z1],
[LR7]  506:              schrift.kopf.offset*z2,0);
[SU7]  507:   schrift.kopf.zeilen:=z1;
[L4L]  508:   schrift.kopf.kursiv_anz:=succ(schrift.kop
       f.zeilen) shr 1 +2;
[Y45]  509:   savechar(zeich);
[BO3]  510:   mouseon;
[HB3]  511:   bildaufbau;
[ZN5]  512:   writechar(zeich);
[TUO]  513: end;
[000]  514:
[H47]  515: procedure spaltenanz;
[EMO]  516: var
[CK4]  517:   z1,z2 : integer;
[BH1]  518: begin
[EO3]  519:   mouseoff;
[X52]  520:   repeat
[WF5]  521:     z1:=lesewert(14,17,3);
[ZR5]  522:   until (z1>0) and (z1<121);
[L38]  523:   z2:=abs(zeichen_breite-z1);
[8TN]  524:   if zeichen_breite<z1 then insertbit(z2,ze
       ich) else deletebit(z2,zeich);
```

```
[Y45]  525:    savechar(zeich);
[B03]  526:    mouseon;
[HB3]  527:    bildaufbau;
[ZN5]  528:    writechar(zeich);
[TUO]  529:  end;
[OOO]  530:
[K7A]  531:  function bestaetigung:boolean;
[EMO]  532:  var
[AF2]  533:    i     : byte;
[5Z4]  534:    xm,ym : integer;
[BH1]  535:  begin
[E03]  536:    mouseoff;
[157]  537:    move(screen,screen1,16384);
[E9A]  538:    gotoxy(16,10);for i:=1 to 48 do write(#32
       );
[ORM]  539:    gotoxy(16,11);write(' Wollen Sie den FONT
       Editor wirklich verlassen ? ');
[P9A]  540:    gotoxy(16,12);for i:=1 to 48 do write(#32
       );
[8ZB]  541:    gotoxy(16,13);write('            JA
                    NEIN          ');
[Y9A]  542:    gotoxy(16,14);for i:=1 to 48 do write(#32
       );
[A84]  543:    draw(122,73,501,73,1);
[VK4]  544:    draw(501,73,501,110,1);
[OF4]  545:    draw(501,110,122,110,1);
[ZH4]  546:    draw(122,110,122,73,1);
[B03]  547:    mouseon;
[X52]  548:    repeat
[D28]  549:      repeat until not lbutton;
[W96]  550:      repeat until lbutton;
[H63]  551:      xm:=xmouse;
[R63]  552:      ym:=ymouse;
[FTD]  553:    until ((ym div 8) in [12,13,14]) and (xm>
       122) and (xm<501);
[BZ9]  554:    if xm>311 then bestaetigung:=false
[N68]  555:              else bestaetigung:=true;
[E03]  556:    mouseoff;
[EZ6]  557:    move(screen1,screen,16384);
[B03]  558:    mouseon;
[TUO]  559:  end;
[OOO]  560:
[146]  561:  procedure zeichnen;
[EMO]  562:  var
[UZ4]  563:    xp,yp : integer;
[NL4]  564:    ende  : boolean;
[BH1]  565:  begin
[863]  566:    ende:=false;
[X52]  567:    repeat
[KNA]  568:      repeat until lbutton or rbutton;
[R63]  569:      xp:=xmouse;
[X63]  570:      yp:=ymouse;
[EF7]  571:      if lbutton and (xp<160) then
[394]  572:        case (yp shr 3) of
[U05]  573:          4 : nameeingabe;
[E25]  574:          5 : fonteingabe;
[CC5]  575:          6 : pointseingabe;
[GXH]  576:          8 : begin delchar(zeich);;bildaufba
       u;writechar(zeich);end;
[5M5]  577:          9 : savechar(zeich);
[R28]  578:          10: savefont(schriftname);
[5CA]  579:          11: if bestaetigung then ende:=true
       ;
[U9F]  580:          13: begin zeichen;bildaufbau;writec
       har(zeich);end;
[ATE]  581:          14: begin   ascii;bildaufbau;writec
       har(zeich);end;
[8A4]  582:          15: zeilenanz;
[WZ4]  583:          16: spaltenanz;
[TUO]  584:        end;
[KAB]  585:      if (xp>159) and (xp<166) and lbutton th
       en
[1ME]  586:        if ((yp-012) div 4)=schrift.kopf.ober
       laenge then
[PZ4]  587:          while lbutton do
[BH1]  588:          begin
[GC5]  589:            yp:=(ymouse-12) div 4;
[4L9]  590:            if yp<(schrift.kopf.oberlaenge+
[EGA]  591:                schrift.kopf.unterlaenge) then
[BH1]  592:            begin
[OBB]  593:              delmarker(schrift.kopf.oberlaen
       ge);
[FOH]  594:              schrift.kopf.unterlaenge:=schri
       ft.kopf.unterlaenge+
[9C8]  595:                                       schri
       ft.kopf.oberlaenge-
[AGO]  596:                                       yp;
[M79]  597:              schrift.kopf.oberlaenge:=yp;
[7CB]  598:              setmarker(schrift.kopf.oberlaen
       ge);
[TUO]  599:            end;
[DLO]  600:          end
[A11]  601:        else
[PTC]  602:         if ((yp-012) div 4)=schrift.kopf.obe
       rlaenge+
[POA]  603:                                  schrift.kopf
       .unterlaenge then
[PZ4]  604:          while lbutton do
[BH1]  605:          begin
```

```
[GC5]  606:            yp:=(ymouse-12) div 4;
[5PA]  607:            if yp>schrift.kopf.oberlaenge the
       n
[BH1]  608:            begin
[SPL]  609:              delmarker(schrift.kopf.oberlaen
       ge+schrift.kopf.unterlaenge);
[VUH]  610:              schrift.kopf.unterlaenge:=yp-sc
       hrift.kopf.oberlaenge;
[ZRL]  611:              setmarker(schrift.kopf.oberlaen
       ge+schrift.kopf.unterlaenge);
[TUO]  612:            end;
[TUO]  613:          end;
[2E6]  614:      xp:=succ((xp-167) shr 2);
[PC6]  615:      yp:=succ((yp-007) shr 2);
[A1A]  616:      if  (xp>0) and (xp<=zeichen_breite)
[U1E]  617:      and (yp>0) and (yp<=schrift.kopf.zeilen
       ) then
[BH1]  618:      begin
[CTA]  619:        if lbutton then setpoint(xp,yp);
[Y7B]  620:        if rbutton then resetpoint(xp,yp);
[TUO]  621:      end;
[FY2]  622:    until ende;
[TUO]  623:  end;
[OOO]  624:
[BH1]  625:  begin
[5K5]  626:    if mouseinit then
[BH1]  627:    begin
[W71]  628:      if paramcount>0 then schriftname:=param
       str(1) else (* Diese Zeile muß bei Turbo Pas
       cal 2.x entfernt werden. *)
[RDS]  629:      begin
                   (* Diese Zeile muß bei Turbo Pas
       cal 2.x entfernt werden. *)
[B27]  630:        writeln(prgname,' : ');
[AZ2]  631:        lowvideo;
[4MF]  632:        write('Bitte geben Sie einen Schriftd
       ateinamen ein : ');
[U16]  633:        readln(schriftname);
[V53]  634:        normvideo;
[5WR]  635:      end;
                   (* Diese Zeile muß bei Turbo Pas
       cal 2.x entfernt werden. *)
[VM9]  636:      if loadfont(schriftname) then
[BH1]  637:      begin
[2L2]  638:        zeich:=#32;
[5Y1]  639:        hires;
[B03]  640:        mouseon;
[HB3]  641:        bildaufbau;
[ZN5]  642:        writechar(zeich);
[5X2]  643:        zeichnen;
[E03]  644:        mouseoff;
[813]  645:        textmode;
[MWN]  646:      end else writeln('Die Schriftdatei konn
       te nicht gefunden werden.');
[DSK]  647:    end else writeln('Microsoft Mouse wurde n
       icht gefunden !');
[5TO]  648:  end.
```

**Listing 2. Mit »Fonted« können Sie eigene Schriften kreieren**

```
[Y46]    0:  program SystemFont;
[OOO]    1:
[VM1]    2:  const
[ZFD]    3:    z129:array[0..7] of byte = ($66,$00,$c6,$
       c6,$c6,$c6,$7e,$00);
[MFD]    4:    z132:array[0..7] of byte = ($66,$00,$7c,$
       06,$7e,$c6,$7e,$00);
[W2E]    5:    z142:array[0..7] of byte = ($c6,$38,$6c,$
       c6,$fe,$c6,$c6,$00);
[6HD]    6:    z148:array[0..7] of byte = ($66,$00,$7c,$
       c6,$c6,$c6,$7c,$00);
[3HD]    7:    z153:array[0..7] of byte = ($c6,$38,$6c,$
       c6,$c6,$6c,$38,$00);
[6FD]    8:    z154:array[0..7] of byte = ($66,$00,$c6,$
       c6,$c6,$c6,$7c,$00);
[56D]    9:    z225:array[0..7] of byte = ($3c,$66,$7c,$
       66,$66,$7c,$60,$60);
[OOO]   10:
[EMO]   11:  var
[E3G]   12:    charbyte     : array[0..127,0..7] of byte
        absolute $ff00:$a6e;
[9N7]   13:    schriftdatei : file of byte;
[737]   14:    schriftname  : string[80];
[575]   15:    i,j,k        : integer;
[OOO]   16:
[3G8]   17:  procedure schreibe(b:byte);
[BH1]   18:  begin
[CW4]   19:    k:=succ(k) mod 40;
[VP5]   20:    if k=0 then write(#46);
[8U6]   21:    write(schriftdatei,b);
[TUO]   22:  end;
[OOO]   23:
[BH1]   24:  begin
[XVO]   25:    k:=0;
[WUC]   26:    if paramcount=0 then schriftname:='SFont'
[4D9]   27:                    else schriftname:=paramst
       r(1);
```

```
[SHF]   28:   writeln('SystemFONT - written by Ramon Ke
          tzlick');
[AZ2]   29:   lowvideo;
[9Z2]   30:   writeln;
[9LW]   31:   writeln('Dieses Programm erstellt eine Sy
          stem-Schrift für TurboDTP und den FontEditor
           :');
[XEC]   32:   assign(schriftdatei,schriftname+'.FNT');
[KG7]   33:   rewrite(schriftdatei);
[B1A]   34:   write('- Schreibe Schriftdaten');
[15C]   35:   schreibe(1); schreibe(0);       (* FONT-N
          ummer *)
[A7C]   36:   schreibe(8); schreibe(0);       (* Point-
          Wert *)
[HEA]   37:   for i:=1 to 5 do                (* Schrif
          tname *)
[BXC]   38:     if i>length(schriftname) then schreibe(
          32)
[VWA]   39:                                   else schreibe(
          ord(schriftname[i]));
[PDA]   40:   for i:=1 to 31 do schreibe(0);  (* Kopf 1
           *)
[24C]   41:   schreibe(6);  schreibe(0);      (* Oberla
          enge *)
[X8B]   42:   for i:=1 to 6  do schreibe(0);  (* Werte
          1,2,3 *)
[YKC]   43:   schreibe(3);  schreibe(0);      (* Unterl
          aenge *)
[NMB]   44:   for i:=1 to 6 do schreibe(0);   (* Werte
          4,5,6 *)
[G7D]   45:   schreibe(6);  schreibe(0);      (* Kursiv
          -Anzahl *)
[3MC]   46:   schreibe(1);  schreibe(0);      (* Fett-A
          nzahl *)
[SMC]   47:   schreibe(1);  schreibe(0);      (* Unters
          trich *)
[REA]   48:   for i:=1 to 18 do schreibe(0);  (* Kopf 2
           *)
[FUA]   49:   schreibe(224);schreibe(0);      (* Offset
           *)
[FMA]   50:   schreibe(8);  schreibe(0);      (* Zeilen
           *)
[BNA]   51:   for i:=1 to 4 do schreibe(0);   (* Kopf 3
           *)
[2N8]   52:   for i:=0 to 224 do              (* Tabell
          e *)
[BH1]   53:   begin
[AL5]   54:     schreibe((i*8) mod 256);
[8M5]   55:     schreibe((i*8) div 256);
[TUO]   56:   end;
[9Z2]   57:   writeln;
[D8B]   58:   write('- Schreibe Zeichensatzdaten');
[MYA]   59:   for j:=0 to 7 do                (* Ze
          ichendaten : *)
[BH1]   60:   begin
[YBE]   61:     for i:=32 to 127 do            (* Ch
          ar 32-127 aus ROM-kopieren *)
[UT7]   62:       schreibe(charbyte[i,j]);
[6S9]   63:     schreibe(0);                   (* Ch
          ar 128 leer *)
[FKA]   64:     schreibe(z129[j]);             (* Ch
          ar 129 : ue *)
[2R9]   65:     schreibe(0);                   (* Ch
          ar 130 leer *)
[7R9]   66:     schreibe(0);                   (* Ch
          ar 131 leer *)
[XFA]   67:     schreibe(z132[j]);             (* Ch
          ar 132 : ae *)
[2FD]   68:     for i:=133 to 141 do schreibe(0); (* Ch
          ar 133-141 leer *)
[5FA]   69:     schreibe(z142[j]);             (* Ch
          ar 142 : Ae *)
[LFD]   70:     for i:=143 to 147 do schreibe(0); (* Ch
          ar 143-147 leer *)
[FKA]   71:     schreibe(z148[j]);             (* Ch
          ar 148 : oe *)
[LGD]   72:     for i:=149 to 152 do schreibe(0); (* Ch
          ar 149-152 leer *)
[5GA]   73:     schreibe(z153[j]);             (* Ch
          ar 153 : Oe *)
[NGA]   74:     schreibe(z154[j]);             (* Ch
          ar 154 : Ue *)
[EGD]   75:     for i:=155 to 224 do schreibe(0); (* Ch
          ar 155-224 leer *)
[NKA]   76:     schreibe(z225[j]);             (* Ch
          ar 225 : sz *)
[BGD]   77:     for i:=226 to 255 do schreibe(0); (* Ch
          ar 226-255 leer *)
[TUO]   78:   end;
[236]   79:   close(schriftdatei);
[9Z2]   80:   writeln;
[RNK]   81:   writeln('- System-Schrift "',schriftname,
          '" gespeichert.');
[9Z2]   82:   writeln;
[5TO]   83: end.
```

**Listing 3. »Sysfont« erzeugt für Sie eine Schriftdatei**

```
[TC3]    0: (* $C-,K-,I-,R-,V- *)
[XX5]    1: program Snapshot;
[000]    2:
[A21]    3: type
[EGK]    4:   regtyp = record ax,bx,cx,dx,bp,si,di,ds,e
           s,fl : integer; end;
[000]    5:
[VM1]    6: const
[2D8]    7:   kennung1   : integer    = $5b5b;
[8A8]    8:   kennung2   : integer    = $b5b5;
[R68]    9:   aktiv      : boolean    = false;
[C79]   10:   request    : boolean    = false;
[P69]   11:   diskaktiv  : boolean    = false;
[D89]   12:   videoaktiv : boolean    = false;
[HY6]   13:   tsrdseg    : integer    = 0;
[DO7]   14:   tsrsseg    : integer    = 0;
[SZ6]   15:   dossseg    : integer    = 0;
[Y17]   16:   dossofs    : integer    = 0;
[V57]   17:   biosprtsc  : integer    = 0;
[1B7]   18:   biosprtscs : integer    = 0;
[R37]   19:   biostimer  : integer    = 0;
[X87]   20:   biostimers : integer    = 0;
[XY6]   21:   biosfrei   : integer    = 0;
[G17]   22:   biosfreis  : integer    = 0;
[ZZ6]   23:   biosdisk   : integer    = 0;
[K27]   24:   biosdisks  : integer    = 0;
[Z17]   25:   biosvideo  : integer    = 0;
[577]   26:   biosvideos : integer    = 0;
[127]   27:   doserror   : integer    = 0;
[M47]   28:   doserrors  : integer    = 0;
[LZ6]   29:   indosoff   : integer    = 0;
[4O7]   30:   indosseg   : integer    = 0;
[ER7]   31:   bildname   : string[70] = '';
[CZ4]   32:   i          : integer    = 0;
[DZ4]   33:   j          : integer    = 0;
[V45]   34:   dz         : integer    = 0;
[XZ4]   35:   z          : integer    = 0;
[YO5]   36:   y1         : integer    = 0;
[O15]   37:   y2         : integer    = 0;
[XO5]   38:   x1         : integer    = 0;
[ZO5]   39:   x2         : integer    = 0;
[TB4]   40:   x          : byte       = 0;
[UB4]   41:   y          : byte       = 0;
[ZC6]   42:   zeilanz    : byte       = 0;
[5C4]   43:   b1         : byte       = 0;
[7C4]   44:   b2         : byte       = 0;
[C89]   45:   mstatus    : boolean    = false;
[TCH]   46:   reg        : regtyp = (ax:0;bx:0;cx:0;dx:
          0;bp:0;si:0;di:0;ds:0;es:0;fl:0);
[E3D]   47:   imgdatei   : array[1..12] of byte = ($00,
          $01,$00,$08,$00,$01,
[FZ3]   48:                                          $00,
          $02,$00,$A9,$01,$74);
[XME]   49:   gemdatei   : array[1..74] of byte = ($FF,
          $FF,$18,$00,$65,$00,$02,$00,
[OX4]   50:                                          $00,
          $00,$00,$00,$00,$00,$00,$00,
[C35]   51:                                          $F0,
          $07,$71,$05,$00,$00,$C7,$00,
[1O5]   52:                                          $7F,
          $02,$00,$00,$01,$00,$00,$00,
[OX4]   53:                                          $00,
          $00,$00,$00,$00,$00,$00,$00,
[OX4]   54:                                          $00,
          $00,$00,$00,$00,$00,$00,$00,
[CY4]   55:                                          $05,
          $00,$02,$00,$00,$00,$17,$00,
[OX4]   56:                                          $00,
          $00,$00,$00,$00,$00,$00,$00,
[9X4]   57:                                          $01,
          $00,$01,$00,$00,$00,$01,$00,
[691]   58:                                          $01,
          $00);
[000]   59:
[EMO]   60: var
[S4A]   61:   prtscint   : integer absolute $0:$14;
[V3A]   62:   timerint   : integer absolute $0:$20;
[3N8]   63:   int10      : integer absolute $0:$40;
[YV8]   64:   int13      : integer absolute $0:$4c;
[AP8]   65:   int24      : integer absolute $0:$90;
[PU8]   66:   int28      : integer absolute $0:$a0;
[2MA]   67:   videomode  : byte absolute $0040:$0049;
[2EF]   68:   tsrkennung : array[1..2] of integer absol
          ute $0000:$03d4;
[6DG]   69:   monitor1   : array[0..99,0..79] of byte a
          bsolute $b800:$0000;
[AKG]   70:   monitor2   : array[0..99,0..79] of byte a
          bsolute $ba00:$0000;
[22A]   71:   screen1    : array[0..3,0..79] of byte;
[92A]   72:   screen2    : array[0..3,0..79] of byte;
[5EA]   73:   diskbuffer : array[0..63] of integer;
[2F7]   74:   bilddatei  : file of byte;
[F86]   75:   altdtaoff  : integer;
[C96]   76:   altdtaseg  : integer;
```

```
[5T5]   77:   altpsp     : integer;
[000]   78:
[ZD5]   79: procedure foto;
[000]   80:
[VT5]   81: procedure print;
[BH1]   82: begin
[3H9]   83:   for i:=1 to length(bildname) do
[3W3]   84:     for j:=0 to 3 do
[BH1]   85:     begin
[WOM]   86:       monitor1[j,pred(i)+pred(x)]:=mem[$ff0
          0:$a6e+ord(bildname[i])*8+j*2];
[ZOM]   87:       monitor2[j,pred(i)+pred(x)]:=mem[$ff0
          0:$a6f+ord(bildname[i])*8+j*2];
[TUO]   88:     end;
[TUO]   89: end;
[000]   90:
[XE6]   91: procedure dateiname;
[BH1]   92: begin
[077]   93:   move(monitor1,screen1,320);
[A77]   94:   move(monitor2,screen2,320);
[M37]   95:   fillchar(monitor1,320,0);
[N37]   96:   fillchar(monitor2,320,0);
[A17]   97:   bildname:='Dateiname : ';
[EWO]   98:   x:=1;
[DZ1]   99:   print;
[T14]  100:   bildname:='_';
[861]  101:   x:=13;
[X52]  102:   repeat
[F37]  103:     bildname:=bildname+#32;
[DZ1]  104:     print;
[L2B]  105:     delete(bildname,length(bildname),1);
[9N2]  106:     reg.ax:=$0;
[AB3]  107:     intr($16,reg);
[2W3]  108:     y:=lo(reg.ax);
[W7C]  109:     if chr(y) in ['0'..'9','a'..'z','A'..'Z
          ','_','\',':'] then
[HF7]  110:       if length(bildname)<67 then
[HHK]  111:         bildname:=copy(bildname,1,pred(leng
          th(bildname)))+chr(y)+'_';
[XMA]  112:     if (y=8) and (length(bildname)>1) then
[71F]  113:       bildname:=copy(bildname,1,length(bild
          name)-2)+'_';
[4Y2]  114:   until y=13;
[L2B]  115:   delete(bildname,length(bildname),1);
[V87]  116:   move(screen1,monitor1,320);
[497]  117:   move(screen2,monitor2,320);
[TUO]  118: end;
[000]  119:
[647]  120: procedure ausschnitt;
[000]  121:
[U76]  122: procedure mouseoff;
[BH1]  123: begin
[WG2]  124:   reg.ax:=2;
[SB3]  125:   intr($33,reg);
[TUO]  126: end;
[000]  127:
[VZ5]  128: procedure mouseon;
[BH1]  129: begin
[VG2]  130:   reg.ax:=1;
[SB3]  131:   intr($33,reg);
[TUO]  132: end;
[000]  133:
[UWA]  134: procedure getpos(var x,y:integer);
[BH1]  135: begin
[XG2]  136:   reg.ax:=3;
[SB3]  137:   intr($33,reg);
[TS4]  138:   x:=reg.cx and $fff8;
[ZF2]  139:   y:=reg.dx;
[TUO]  140: end;
[000]  141:
[3X9]  142: procedure setpos(x,y:integer);
[BH1]  143: begin
[YG2]  144:   reg.ax:=4;
[CL2]  145:   reg.cx:=x;
[KL2]  146:   reg.dx:=y;
[SB3]  147:   intr($33,reg);
[TUO]  148: end;
[000]  149:
[EN6]  150: procedure testmouse;
[BH1]  151: begin
[GL5]  152:   mstatus:=false;
[9T3]  153:   getpos(x1,y1);
[YV3]  154:   for i:=0 to 3 do
[WL3]  155:     for j:=0 to 31 do
[6CH]  156:       if odd(j) then screen1[i,j]:=monitor2
          [(y1+j-16) div 2,x1 div 8 +i-2]
[5WE]  157:                 else screen1[i,j]:=monitor1
          [(y1+j-16) div 2,x1 div 8 +i-2];
[EO3]  158:   mouseoff;
[YV3]  159:   for i:=0 to 3 do
[WL3]  160:     for j:=0 to 31 do
[BH1]  161:     begin
[LAE]  162:       if odd(j) then b1:=monitor2[(y1+j-16)
          div 2,x1 div 8 +i-2]
[G4D]  163:                 else b1:=monitor1[(y1+j-16)
          div 2,x1 div 8 +i-2];
[W8C]  164:       if b1<>screen1[i,j] then mstatus:=tru
          e;
[TUO]  165:     end;
[TUO]  166: end;
[000]  167:
[UU8]  168: function button:boolean;
[BH1]  169: begin
[OL4]  170:   button:=false;
[XG2]  171:   reg.ax:=3;
[SB3]  172:   intr($33,reg);
[WRB]  173:   if (reg.bx and 1)=1 then button:=true;
[TUO]  174: end;
[000]  175:
[LYC]  176: procedure xordraw(xa,ya,xb,yb:integer);
[000]  177:
[GCA]  178: procedure xorplot(x,y:integer);
[BH1]  179: begin
[HF4]  180:   b1:=$80 shr (x mod 8);
[HHA]  181:   if (x>=0) and (x<=639) and (y>=0) and (y<
          =199) then
[O8N]  182:     if odd(y) then monitor2[y div 2,x div 8
          ]:=monitor2[y div 2,x div 8] xor b1
[78L]  183:               else monitor1[y div 2,x div 8
          ]:=monitor1[y div 2,x div 8] xor b1;
[TUO]  184: end;
[000]  185:
[BH1]  186: begin
[AO4]  187:   if xa=xb then
[BH1]  188:   begin
[GL4]  189:     dz:=abs(yb-ya);
[RL5]  190:     if ya<yb then z:=ya
[AT2]  191:              else z:=yb;
[P33]  192:     if dz>0 then
[OL3]  193:       while dz>0 do
[BH1]  194:       begin
[3H4]  195:         xorplot(xa,z);
[2P3]  196:         dz:=pred(dz);
[DW2]  197:         z:=succ(z)
[TUO]  198:       end;
[LM2]  199:   end else
[BH1]  200:   begin
[BL4]  201:     dz:=abs(xb-xa);
[AL5]  202:     if xa<xb then z:=xa
[9T2]  203:              else z:=xb;
[P33]  204:     if dz>0 then
[OL3]  205:       while dz>0 do
[BH1]  206:       begin
[YH4]  207:         xorplot(z,ya);
[2P3]  208:         dz:=pred(dz);
[DW2]  209:         z:=succ(z)
[TUO]  210:       end;
[TUO]  211:   end;
[TUO]  212: end;
[000]  213:
[CV5]  214: procedure kreuz;
[BH1]  215: begin
[SP6]  216:   xordraw(x2-8,y2,x2+8,y2);
[AD6]  217:   xordraw(x2,y2-5,x2,y2+5);
[TUO]  218: end;
[000]  219:
[GS5]  220: procedure kasten;
[BH1]  221: begin
[XH6]  222:   xordraw(x1  ,y1,x2+7,y1);
[3P6]  223:   xordraw(x2+7,y1,x2+7,y2);
[BK6]  224:   xordraw(x2+7,y2,x1  ,y2);
[5E6]  225:   xordraw(x1  ,y2,x1  ,y1);
[TUO]  226: end;
[000]  227:
[BH1]  228: begin
[G93]  229:   testmouse;
[XG2]  230:   reg.ax:=3;
[SB3]  231:   intr($33,reg);
[FF2]  232:   i:=reg.cx;
[GF2]  233:   j:=reg.dx;
[1H2]  234:   reg.ax:=7;
[4H2]  235:   reg.cx:=0;
[PX2]  236:   reg.dx:=639;
[SB3]  237:   intr($33,reg);
[2H2]  238:   reg.ax:=8;
[4H2]  239:   reg.cx:=0;
[XX2]  240:   reg.dx:=199;
[SB3]  241:   intr($33,reg);
[RU3]  242:   setpos(320,100);
[ET3]  243:   getpos(x2,y2);
[PO2]  244:   kreuz;
[X52]  245:   repeat
[PO2]  246:     kreuz;
[ET3]  247:     getpos(x2,y2);
[PO2]  248:     kreuz;
[3V3]  249:   until button;
[PO2]  250:   kreuz;
[XW1]  251:   x1:=x2;
[3X1]  252:   y1:=y2;
[ET3]  253:   getpos(x2,y2);
[OM2]  254:   kasten;
[X52]  255:   repeat
[OM2]  256:     kasten;
[ET3]  257:     getpos(x2,y2);
[8T5]  258:     if x2<x1 then x2:=x1;
[XT5]  259:     if y2<y1 then y2:=y1;
```

```
[OM2] 260:      kasten;
[XG5] 261:    until not button;
[OM2] 262:    kasten;
[PM2] 263:    x1:=x1 div 8;
[YM2] 264:    x2:=x2 div 8;
[N53] 265:    setpos(i,j);
[5F8] 266:    if mstatus then mouseon;
[TUO] 267: end;
[000] 268:
[C68] 269: procedure dateioeffnen;
[000] 270:
[K46] 271: procedure posdaten;
[BH1] 272: begin
[E2D] 273:    b1:=lo(x1*8);  b2:=hi(x1*8);  write(bildd
      atei,b1,b2);
[FTC] 274:    b1:=lo(y1);    b2:=hi(y1);    write(bildd
      atei,b1,b2);
[VFD] 275:    b1:=lo(x2*8+7);b2:=hi(x2*8+7);write(bildd
      atei,b1,b2);
[GTC] 276:    b1:=lo(y2);    b2:=hi(y2);    write(bildd
      atei,b1,b2);
[TUO] 277: end;
[000] 278:
[BH1] 279: begin
[0K3] 280:    reg.ax:=$2f00;
[KZ2] 281:    msdos(reg);
[4D5] 282:    altdtaoff:=reg.bx;
[WD5] 283:    altdtaseg:=reg.es;
[7D3] 284:    reg.ax:=$5100;
[KZ2] 285:    msdos(reg);
[A25] 286:    altpsp:=reg.bx;
[FH3] 287:    reg.ax:=$1a00;
[1N3] 288:    reg.ds:=dseg;
[0H7] 289:    reg.dx:=ofs(diskbuffer);
[KZ2] 290:    msdos(reg);
[4D3] 291:    reg.ax:=$5000;
[MN3] 292:    reg.bx:=cseg;
[KZ2] 293:    msdos(reg);
[XZ9] 294:    assign(bilddatei,bildname+'.GEM');
[B66] 295:    rewrite(bilddatei);
[PFD] 296:    for i:=1 to 8 do write(bilddatei,gemdatei
      [i]);
[KY2] 297:    posdaten;
[P7E] 298:    for i:=17 to 52 do write(bilddatei,gemdat
      ei[i]);
[BG7] 299:    bildname:=bildname+'.IMG';
[OW0] 300:    j:=1;
[C5M] 301:    for i:=1 to length(bildname) do if bildna
      me[i]='\' then j:=succ(i);
[TU7] 302:    b1:=lo(length(bildname)+6-j);
[3T7] 303:    b2:=hi(length(bildname)+6-j);
[ZX6] 304:    write(bilddatei,b1,b2);
[G8E] 305:    for i:=55 to 56 do write(bilddatei,gemdat
      ei[i]);
[KY2] 306:    posdaten;
[E8E] 307:    for i:=65 to 74 do write(bilddatei,gemdat
      ei[i]);
[U51] 308:    b2:=0;
[3H9] 309:    for i:=j to length(bildname) do
[BH1] 310:    begin
[ML8] 311:      b1:=ord(upcase(bildname[i]));
[ZX6] 312:      write(bilddatei,b1,b2);
[TUO] 313:    end;
[822] 314:    b1:=$ff;
[UX6] 315:    write(bilddatei,b1,b1);
[BC5] 316:    close(bilddatei);
[2E8] 317:    assign(bilddatei,bildname);
[B66] 318:    rewrite(bilddatei);
[SYD] 319:    for i:=1 to 12 do write(bilddatei,imgdate
      i[i]);
[444] 320:    b1:=hi(x2*8-x1*8+8);
[Y54] 321:    b2:=lo(x2*8-x1*8+8);
[ZX6] 322:    write(bilddatei,b1,b2);
[ED4] 323:    b1:=hi(succ(y2-y1));
[8F4] 324:    b2:=lo(succ(y2-y1));
[ZX6] 325:    write(bilddatei,b1,b2);
[TUO] 326: end;
[000] 327:
[FT5] 328: procedure point;
[BH1] 329: begin
[HU3] 330:    if x<=x2 then
[BH1] 331:    begin
[V41] 332:      i:=x;
[X52] 333:      repeat
[WO3] 334:        i:=succ(i);
[TZR] 335:        if odd(y) then begin;b1:=monitor2[y d
      iv 2,i];b2:=monitor2[y div 2,succ(i)];end
[RSK] 336:                  else begin;b1:=monitor1[y d
      iv 2,i];b2:=monitor1[y div 2,succ(i)];end;
[U4A] 337:      until ((b1=b2) and (b1 in [$00,$ff])) o
      r (i>x2);
[YE6] 338:      if i=x2 then i:=succ(i);
[B62] 339:      b2:=i-x;
[UD1] 340:      b1:=$80;
[ZX6] 341:      write(bilddatei,b1,b2);
[8V3] 342:    for i:=1 to b2 do
[BH1] 343:    begin
[7CC] 344:      if odd(y) then b1:=monitor2[y div 2,x
      ] xor 255
[AYA] 345:                else b1:=monitor1[y div 2,x
      ] xor 255;
[DW5] 346:      write(bilddatei,b1);
[U13] 347:      x:=succ(x);
[TUO] 348:    end;
[TUO] 349:  end;
[TUO] 350: end;
[000] 351:
[A49] 352: procedure testbitwdh(t:byte);
[BH1] 353: begin
[V41] 354:  i:=x;
[X52] 355:  repeat
[UWA] 356:    if odd(y) then b1:=monitor2[y div 2,i]
[DT8] 357:              else b1:=monitor1[y div 2,i];
[KF6] 358:    if b1=t then i:=succ(i);
[146] 359:  until (i>x2) or (b1<>t);
[VM2] 360:  if i>x then
[BH1] 361:  begin
[G52] 362:    case t of
[FC3] 363:      $ff : b1:=i-x;
[3E3] 364:      $00 : b1:=i-x+$80;
[TUO] 365:    end;
[DW5] 366:    write(bilddatei,b1);
[7U0] 367:    x:=i
[TUO] 368:  end;
[TUO] 369: end;
[000] 370:
[ZTD] 371: function verglzeilen(z1,z2:byte):boolean;
[BH1] 372: begin
[RT5] 373:  verglzeilen:=true;
[7E4] 374:  for i:=x1 to x2 do
[BH1] 375:  begin
[PKB] 376:    if odd(z1) then b1:=monitor2[z1 div 2,i
      ]
[6T8] 377:              else b1:=monitor1[z1 div 2,i
      ];
[OLB] 378:    if odd(z2) then b2:=monitor2[z2 div 2,i
      ]
[HT8] 379:              else b2:=monitor1[z2 div 2,i
      ];
[AP9] 380:    if b1<>b2 then verglzeilen:=false;
[TUO] 381:  end;
[TUO] 382: end;
[000] 383:
[CS7] 384: procedure testwdhline;
[BH1] 385: begin
[EA3] 386:  zeilanz:=1;
[RXS] 387:  while ((y+zeilanz)<=y2) and verglzeilen(y
      ,y+zeilanz) do zeilanz:=succ(zeilanz);
[Y25] 388:  if zeilanz>1 then
[BH1] 389:  begin
[AC1] 390:    b1:=$00;
[A22] 391:    b2:=$ff;
[L6A] 392:    write(bilddatei,b1,b1,b2,zeilanz);
[TUO] 393:  end;
[TUO] 394: end;
[000] 395:
[P16] 396: procedure bildende;
[BH1] 397: begin
[AC1] 398:  b1:=$00;
[WB7] 399:  write(bilddatei,b1,b1,b1);
[BC5] 400:  close(bilddatei);
[FH3] 401:  reg.ax:=$1a00;
[PG5] 402:  reg.ds:=altdtaseg;
[RG5] 403:  reg.dx:=altdtaoff;
[KZ2] 404:  msdos(reg);
[4D3] 405:  reg.ax:=$5000;
[A25] 406:  reg.bx:=altpsp;
[KZ2] 407:  msdos(reg);
[TUO] 408: end;
[000] 409:
[BH1] 410: begin
[9X7] 411:  if videomode in [4,5,6] then
[BH1] 412:  begin
[T13] 413:    dateiname;
[2L7] 414:    if length(bildname)>0 then
[BH1] 415:    begin
[YE3] 416:      ausschnitt;
[ZE1] 417:      y:=y1;
[UE1] 418:      x:=x1;
[G64] 419:      dateioeffnen;
[D84] 420:      while y<=y2 do
[BH1] 421:      begin
[AU3] 422:        testwdhline;
[484] 423:        while x<=x2 do
[BH1] 424:        begin
[YA4] 425:          testbitwdh($00);
[GX4] 426:          testbitwdh($ff);
[7Z1] 427:          point;
[TUO] 428:        end;
[5W3] 429:        y:=y+zeilanz;
[UE1] 430:        x:=x1;
[TUO] 431:      end;
[CW2] 432:      bildende;
[TUO] 433:    end;
[TUO] 434:  end;
[TUO] 435: end;
[000] 436:
[MB8] 437: procedure prtschandler;
```

```
[BH1]  438: begin
[ZZ6]  439:   inline($fa/                    (* cli *)
[HG5]  440:           $50/                   (* push ax
       *)
[GZ5]  441:           $b4/$01/               (* mov ah,1
       *)
[V8C]  442:           $2e/$88/$26/request/   (* mov cs:[
       request],ah *)
[E95]  443:           $58/                   (* pop ax *
       )
[317]  444:           $8b/$e5/               (* mov sp,b
       p *)
[TD5]  445:           $5d/                   (* pop bp *
       )
[ED5]  446:           $cf);                  (* iret *)
[TUO]  447: end;
[000]  448:
[EA8]  449: procedure timerhandler;
[BH1]  450: begin
[DR7]  451:   inline($9c/                    (* pushf *)
[N3C]  452:           $2e/$ff/$1e/biostimer);(* call bio
       stimer *)
[CV7]  453:   inline($50/                    (* push ax
       *)
[UG5]  454:           $53/                   (* push bx
       *)
[MG5]  455:           $51/                   (* push cx
       *)
[RG5]  456:           $52/                   (* push dx
       *)
[RG5]  457:           $57/                   (* push di
       *)
[MG5]  458:           $56/                   (* push si
       *)
[MG5]  459:           $06/                   (* push es
       *)
[3N5]  460:           $1e/                   (* push ds
       *)
[705]  461:           $fb);                  (* sti *)
[2H5]  462:   if request then
[TXH]  463:     if (not aktiv) and (not diskaktiv) and
       (not videoaktiv) and
[2SA]  464:        (mem[indosseg:indosoff]=0) then
[BH1]  465:     begin
[383]  466:       aktiv:=true;
[YV2]  467:       inline($fa/
[5RC]  468:             $2e/$8c/$16/dossseg/   (* mov
       cs:[dossseg],ss *)
[K27]  469:             $8c/$d6/               (* mov
       si,ss *)
[KRC]  470:             $2e/$8e/$16/tsrsseg/   (* mov
       ss,cs:[tsrsseg] *)
[7SC]  471:             $2e/$8e/$1e/tsrdseg/   (* mov
       ds,cs:[tsrdseg] *)
[ZLC]  472:             $2e/$3b/$36/tsrsseg/   (* cmp
       si,cs:[tsrsseg] *)
[HY6]  473:             $89/$e6/               (* mov
       si,sp *)
[YR4]  474:             $74/$05/               (* je +
       5 *)
[44N]  475:             $3e/$8b/$36/$2b/$0b/   (* mov
       si,ds:[b2b] ; Die Daten $2b/$0b/ müssen wahr
       scheinlich
[XPE]  476:
                        bei Turbo Pascal 2.x in $74/$0
       1/ geändert werden. *)
[957]  477:             $87/$f4/               (* xchg
       sp,si *)
[MG5]  478:             $56/                   (* push
       si *)
[705]  479:             $fb);                  (* sti
       *)
[3C1]  480:       foto:
[TP3]  481:       aktiv:=false;
[GK5]  482:       request:=false;
[ZZ6]  483:       inline($fa/                  (* cli
       *)
[LH5]  484:             $5e/                   (* pop
       si *)
[MX6]  485:             $89/$f4/               (* mov
       sp,si *)
[NGC]  486:             $2e/$8e/$16/dossseg/   (* mov
       ss,cs:[dossseg] *)
[705]  487:             $fb);                  (* sti
       *)
[TUO]  488:     end;
[HZ7]  489:   inline($1f/                      (* pop ds *
       )
[X95]  490:          $07/                      (* pop es *
       )
[LH5]  491:          $5e/                      (* pop si *
       )
[DE5]  492:          $5f/                      (* pop di *
       )
[YD5]  493:          $5a/                      (* pop dx *
       )
[Z95]  494:          $59/                      (* pop cx *
       )
[LD5]  495:          $5b/                      (* pop bx *
       )
[E95]  496:          $58/                      (* pop ax *
       )
[317]  497:          $8b/$e5/                  (* mov sp,b
       p *)
[TD5]  498:          $5d/                      (* pop bp *
       )
[ED5]  499:          $cf);                     (* iret *)
[TUO]  500: end;
[000]  501:
[YK7]  502: procedure int10handler;
[BH1]  503: begin
[CV7]  504:   inline($50/                      (* push ax
       *)
[GZ5]  505:          $b4/$01/                  (* mov ah,1
       *)
[R1E]  506:          $2e/$88/$26/videoaktiv/(* mov cs:[
       videoaktiv],AH *)
[XC5]  507:          $58);                     (* pop ax *
       )
[DR7]  508:   inline($9c/                      (* pushf *)
[ZOC]  509:          $2e/$ff/$1e/biosvideo);(* call bio
       svideo *)
[CV7]  510:   inline($50/                      (* push ax
       *)
[7Z5]  511:          $b4/$00/                  (* mov ah,0
       *)
[R1E]  512:          $2e/$88/$26/videoaktiv/(* mov cs:[
       videoaktiv],AH *)
[E95]  513:          $58/                      (* pop ax *
       )
[317]  514:          $8b/$e5/                  (* mov sp,b
       p *)
[TD5]  515:          $5d/                      (* pop bp *
       )
[ED5]  516:          $cf);                     (* iret *)
[TUO]  517: end;
[000]  518:
[LL7]  519: procedure int13handler;
[BH1]  520: begin
```


## Eine Bitte an unsere Abonnenten

*Vermerken Sie bei Schriftverkehr und Zahlungen neben der vollständigen Anschrift stets Ihre Abo-Nummer.*

*Sie vermeiden damit unnötige Verzögerungen bei der Bearbeitung Ihres Abonnements.*

**Vielen Dank**
**Ihre DMV-Versandabteilung**

```
[CV7]  521:    inline($50/                    (* push ax
       *)
[GZ5]  522:           $b4/$01/                (* mov ah,1
       *)
[X9D]  523:           $2e/$88/$26/diskaktiv/ (* mov cs:[
       diskaktiv],AH *)
[XC5]  524:           $58):                   (* pop ax *
       )
[DR7]  525:    inline($9c/                    (* pushf *)
[CLB]  526:           $2e/$ff/$1e/biosdisk); (* call bio
       sdisk *)
[CV7]  527:    inline($50/                    (* push ax
       *)
[7Z5]  528:           $b4/$00/                (* mov ah,0
       *)
[X9D]  529:           $2e/$88/$26/diskaktiv/ (* mov cs:[
       diskaktiv],AH *)
[E95]  530:           $58/                    (* pop ax *
       )
[317]  531:           $8b/$e5/                (* mov sp,b
       p *)
[TD5]  532:           $5d/                    (* pop bp *
       )
[ED5]  533:           $cf);                   (* iret *)
[TUO]  534: end;
[OOO]  535:
[OM7]  536: procedure int24handler;
[BH1]  537: begin
[TE8]  538:    inline($b0/$03/                (* mov al,3
       *)
[317]  539:           $8b/$e5/                (* mov sp,b
       p *)
[TD5]  540:           $5d/                    (* pop bp *
       )
[ED5]  541:           $cf);                   (* iret *)
[TUO]  542: end;
[OOO]  543:
[OOO]  544:
[WM7]  545: procedure int28handler;
[BH1]  546: begin
[CV7]  547:    inline($50/                    (* push ax
       *)
[UG5]  548:           $53/                    (* push bx
       *)
[MG5]  549:           $51/                    (* push cx
       *)
[RG5]  550:           $52/                    (* push dx
       *)
[RG5]  551:           $57/                    (* push di
       *)
[MG5]  552:           $56/                    (* push si
       *)
[MG5]  553:           $06/                    (* push es
       *)
[3N5]  554:           $1e/                    (* push ds
       *)
[7O5]  555:           $fb);                   (* sti *)
[2H5]  556:    if request then
[VBK]  557:      if (not aktiv) and (not diskaktiv) and
       (not videoaktiv) then
[BH1]  558:      begin
[383]  559:        aktiv:=true;
[YV2]  560:        inline($fa/
[5RC]  561:                  $2e/$8c/$16/dossseg/    (* mov
       cs:[dossseg],ss *)
[K27]  562:                  $8c/$d6/                (* mov
       si,ss *)
[KRC]  563:                  $2e/$8e/$16/tsrsseg/    (* mov
       ss,cs:[tsrsseg] *)
[7SC]  564:                  $2e/$8e/$1e/tsrdseg/    (* mov
       ds,cs:[tsrdseg] *)
[ZLC]  565:                  $2e/$3b/$36/tsrsseg/    (* cmp
       si,cs:[tsrsseg] *)
[HY6]  566:                  $89/$e6/                (* mov
       si,sp *)
[YR4]  567:                  $74/$05/                (* je +
       5 *)
[44N]  568:                  $3e/$8b/$36/$2b/$0b/    (* mov
       si,ds:[b2b] ; Die Daten $2b/$0b/ müssen wahr
       scheinlich
[XPE]  569:
                             bei Turbo Pascal 2.x in $74/$0
       1/ geändert werden. *)
[957]  570:                  $87/$f4/                (* xchg
       sp,si *)
[MG5]  571:                  $56/                    (* push
       si *)
[7O5]  572:                  $fb);                   (* sti
       *)
[3C1]  573:        foto;
[TP3]  574:        aktiv:=false;
[GK5]  575:        request:=false;
[ZZ6]  576:        inline($fa/                       (* cli
       *)
[LH5]  577:                  $5e/                    (* pop
       si *)
[MX6]  578:                  $89/$f4/                (* mov
       sp,si *)
[NGC]  579:                  $2e/$8e/$16/dossseg/    (* mov
       ss,cs:[dossseg] *)
```

```
[7O5]  580:                  $fb);                   (* sti
       *)
[TUO]  581:      end;
[HZ7]  582:    inline($1f/                           (* pop ds *
       )
[X95]  583:           $07/                           (* pop es *
       )
[LH5]  584:           $5e/                           (* pop si *
       )
[DE5]  585:           $5f/                           (* pop di *
       )
[YD5]  586:           $5a/                           (* pop dx *
       )
[Z95]  587:           $59/                           (* pop cx *
       )
[LD5]  588:           $5b/                           (* pop bx *
       )
[E95]  589:           $58/                           (* pop ax *
       )
[317]  590:           $8b/$e5/                       (* mov sp,b
       p *)
[TD5]  591:           $5d/                           (* pop bp *
       )
[ZYB]  592:           $2e/$ff/$2e/biosfrei); (* jmp cs:b
       iosfrei *)
[TUO]  593: end;
[OOO]  594:
[K6M]  595: procedure chint(var interupt,altint:integer
       ;neuint:integer);
[BH1]  596: begin
[5T5]  597:    altint:=interupt;
[BDL]  598:    memw[cseg:ofs(altint)+2]:=memw[seg(interu
       pt):ofs(interupt)+2];
[BY5]  599:    interupt:=neuint;
[FRD]  600:    memw[seg(interupt):ofs(interupt)+2]:=cseg
       ;
[TUO]  601: end;
[OOO]  602:
[BH1]  603: begin
[UG2]  604:    reg.ax:=0;
[SB3]  605:    intr($33,reg);
[236]  606:    if reg.ax<>$ffff then
[BH1]  607:    begin
[R6L]  608:      writeln('Snapshot : Microsoft Mouse wur
       de nicht gefunden.');
[L2H]  609:      writeln('            Das Programm wurde
       nicht installiert.');
[LM2]  610:    end else
[VRL]  611:    if (tsrkennung[1]=kennung1) and (tsrkennu
       ng[2]=kennung2) then
[BH1]  612:    begin
[PTL]  613:      writeln('Snapshot : Das Programm wurde
       bereits installiert.');
[ARK]  614:      writeln('           Aufruf mit [Shift]
       & [PrtSc] im Grafikmodus.');
[LM2]  615:    end else
[BH1]  616:    begin
[5S4]  617:      tsrdseg:=dseg;
[ZU4]  618:      tsrsseg:=sseg;
[CD3]  619:      reg.ax:=$3400;
[KZ2]  620:      msdos(reg);
[MC5]  621:      indosseg:=reg.es;
[EB5]  622:      indosoff:=reg.bx;
[S23]  623:      inline($fa);
[U9E]  624:      chint(prtscint,biosprtsc,ofs(prtschandl
       er));
[O5E]  625:      chint(timerint,biostimer,ofs(timerhandl
       er));
[5CC]  626:      chint(int10   ,biosvideo,ofs(int10handl
       er));
[GEC]  627:      chint(int13   ,biosdisk ,ofs(int13handl
       er));
[TXZ]  628:      chint(int24   ,doserror ,ofs(int24handl
       er)); (* Diese Zeile muß bei MS-DOS kleiner
        als 3.x entfernt werden *)
[ZDC]  629:      chint(int28   ,biosfrei ,ofs(int28handl
       er));
[U23]  630:      inline($fb);
[9OF]  631:      writeln('Snapshot - written by Ramon Ke
       tzlick :');
[AZ2]  632:      lowvideo;
[6HH]  633:      writeln('- Aufruf mit [Shift] & [PrtSc]
        im Grafikmodus.');
[UG7]  634:      tsrkennung[1]:=kennung1;
[5H7]  635:      tsrkennung[2]:=kennung2;
[3D3]  636:      reg.ax:=$3100;
[P1P]  637:      reg.dx:=$800; (* Falls das Programm abs
       türtz (z.B. Speichzuweisungsfehler)
[HEL]  638:                    sollte der Wert $800 v
       ergrößert werden (der Wert muß größer
[X1F]  639:                    als die Länge der COM-
       Datei geteilt durch 16 sein). *)
[KZ2]  640:      msdos(reg);
[TUO]  641:    end;
[5TO]  642: end.
```

**Listing 4. Mit »Snapshot« können Sie Bildschirminhalte in SDTP einbinden**

# DOS-EXTRA
# DATABOX

**– Die Software zur DOS-EXTRA Nr. 4 –**

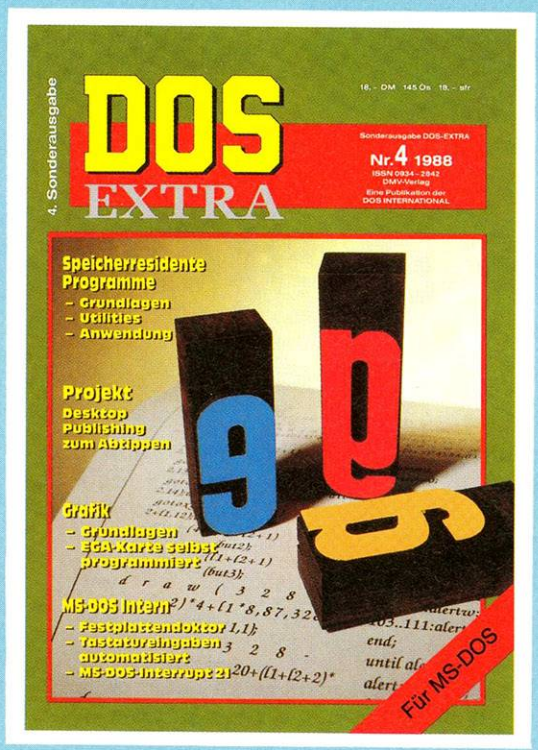
## auf 2 Disketten im Format 5 1/4“ oder 1 Diskette im Format 3 1/2“

**sagenhaft günstig, extra viel gute Software für’s Geld!**

**DATABOX:**

- mehr als der übliche Softwareservice
- bringt alle Programme der DOS EXTRA auf Datenträger

**Für alle PCs unter MS-DOS als 5 1/4“ - oder 3 1/2“ -Diskette**




**Databoxinhalte der DOS EXTRA Nr. 4**

- Desktop-Publishing-Programm »SDTP«
- Kommandoeditor für MS-DOS
- Editor für Disketten und Festplatten
- Speicherresidente Programme auflisten und entfernen
- Paßwort-Schutz für MS-DOS
- Tastatureingabe automatisiert
- Clock-Utility für alle Grafikkarten
- Cursor und Zeichensatz für EGA-Karte programmieren
- Fenstertechnik für Turbo C
- Generator für Hilfsbildschirme
- Geprüfte Eingabe mit »SCheck«

**2 Disketten 5 1/4“** (Bestell-Nr. 3811)
**1 Diskette 3 1/2“** (Bestell-Nr. 3812)

**DM 35, – ***

## Noch erhältich:

**Databox DOS EXTRA Nr. 1**

- SAM, der DOS-Assembler
- Clock, Uhr-Utilities
- Pascal – Include-Datei zum Aufruf von BIOS-Funktionen unter Turbo Pascal
- Tips und Tricks für jeden Zweck

**Diskette 5 1/4“** (Bestell-Nr. 350)
**Diskette 3 1/2“** (Bestell-Nr. 351)

**DM 24, – ***

**Databox DOS EXTRA Nr. 2**

- DOS-Commander, eine Benutzeroberfläche für MS-DOS
- SCopy, ein professionelles Kopierprogramm
- SForth 3.1, ein Compiler im Selbstbau
- SCAD, 3D-Grafik selbstgemacht
- Hurrican, ein Actionspiel der Superlative
- Supermind, ein Strategiespiel
- Utilities, Routinen für die tägliche Arbeit
- SCheck – Checksummer für die Eingabe

**4 Disketten 5 1/4“** (Bestell-Nr. 352)
**2 Disketten 3 1/2“** (Bestell-Nr. 353)

**DM 45, – ***

**Databox DOS EXTRA Nr. 3**

- Datenfernübertragungsprogramm in 100% Assemblercode
- Herkules-Grafik-Toolbox in Turbo Pascal
- Grafikfunktionen für Turbo C
- Zeichensatzeditor für Epson-kompatible Drucker
- Treiber für AT-Tastatur
- alle begleitenden Listings zum Kurs »PC-glasklar«
- Demo für Sprachdigitalisierung
- ...und alle weiteren Listings

**2 Disketten 5 1/4“** (Bestell-Nr. 357)
**1 Diskette 3 1/2“** (Bestell-Nr. 358)

**DM 35, – ***

* Unabhängig von der Anzahl der bestellten Programme berechnen wir für das Inland 3,– DM bzw. für das Ausland 5,– DM Porto und Verpackung. Bitte benutzen Sie die Bestellkarte.

**DMV-Verlag · Postfach 250 · 3440 Eschwege**

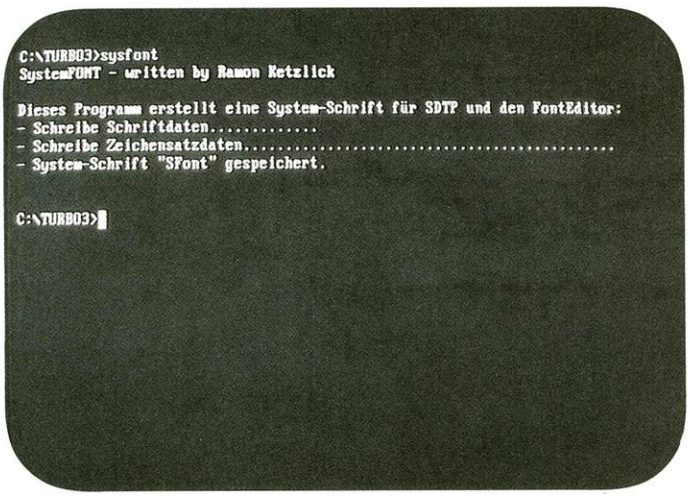



Bild 1. Die Meldung von »sysfont.com«

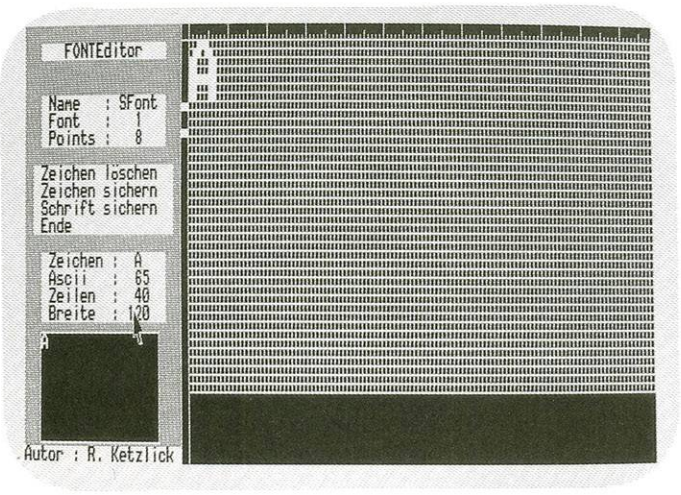



Bild 2. Der Fonteditor

# Layouts bearbeiten mit SDTP

**Wenn Sie auf herkömmliche Weise eine Seite montieren, sind immer wieder bestimmte Arbeitsgänge nötig. Dies gilt auch für den Fall, daß Sie solche Arbeiten mit dem Computer lösen wollen. Wie lassen sich beispielsweise Grafiken integrieren oder Rahmen ziehen?**

Wenn Sie mit SDTP arbeiten wollen, stehen Ihnen zwei Alternativen zur Verfügung. Sie können entweder die Schnappschußfunktion und die Schriftdateien von GEM verwenden oder aber Schriftdateien mit »Sysfont« und »Fonted« selbst erzeugen. Wollen Sie Bildschirminhalte in das Layout einbinden, so ermöglicht Ihnen dies darüber hinaus das Programm Snapshot.
Sollten Sie nicht über GEM verfügen, müssen Sie folgendermaßen vorgehen:
Laden Sie das MS-DOS-Hilfsprogramm »graftabl.exe«. Geben Sie auf der MS-DOS-Betriebssystemebene »sysfont« ein. Dieses Programm erzeugt die Systemschrift für SDTP und speichert die Schriftdatei »sfont.fnt« (Bild 1) auf Ihrer Diskette oder in Ihrem aktuellen Unterverzeichnis. Geben Sie jetzt den Befehl

```
copy sfont.fnt neueschr.fnt
```

ein. Auf Ihrer Diskette befinden sich jetzt zwei Schriftdateien: »sfont.fnt« und «neueschr.fnt«. Laden Sie jetzt das Programm »fonted«, indem Sie

```
fonted neueschr
```

eingeben.
Auf dem Bildschirm sehen Sie jetzt den Fonteditor (Bild 2). Im rechten Teil des Bildschirms wird auf weißem Hintergrund ein Raster ausgegeben, in dem Sie Ihre Buchstaben gestalten können.

Auf der linken Seite sind mehrere Menüpunkte aufgelistet. Der erste Menüteil gibt Ihnen Informationen über den für Sie gerade aktuellen Schriftsatz. Der mittlere Teilbereich dient dem Löschen und Speichern, während der letzte Menüteil angibt, welches Zeichen Sie gerade bearbeiten, die Anzahl der Zeilen und die Breite pro Zeichen. Bewegen Sie jetzt mit der Maus den Cursor zu »Zeichen« und klicken Sie die linke Maustaste. Es erscheint der Unterstrich. Geben Sie jetzt über die Tastatur den Buchstaben ein. Im Raster wird der Buchstabe sichtbar, und im weißen Feld links unten wird der Buchstabe in seiner tatsächlichen Größe gezeigt. Gehen Sie jetzt zu »Zeilen« und klicken Sie erneut mit der linken Maustaste. Wieder erscheint der Unterstrich. Geben Sie jetzt für die 8 eine 12 (das Maximum ist 48 Zeilen) ein. Das Raster wird neuerlich aufgebaut, ist jetzt aber länger. Wenn Sie nun wieder mit der Maus »Zeichen« anklicken und einen anderen Buchstaben eingeben, werden Sie bemerken, daß für jeden Buchstaben das neue Raster angepaßt ist. Bewegen Sie jetzt den Mauscursor zu »Breite« und geben Sie den Wert 32 ein. Das Raster ist breiter geworden. Wenn Sie jetzt einen anderen Buchstaben bei »Zeichen« eingeben, bemerken Sie, daß Sie bei diesem Buchstaben wieder die alte Breite vorfinden. Die Ursache dafür liegt darin, daß – für die Erzeugung eines schönen Schriftbildes – bestimmte Buchstaben, wie zum Beispiel »M«, mehr Platz benötigen.

Jeden Buchstaben, den Sie bearbeitet haben, schließen Sie mit dem Menüpunkt »Zeichen sichern« ab. Ist Ihr neuer Schrifttyp komplett fertiggestellt, muß die Arbeit unbedingt mit »Schrift sichern« beendet werden, denn nur dadurch wird dieser Schrifttyp auf Diskette oder Platte gespeichert. Alles andere spielte sich bisher im Arbeitsspeicher Ihres Computers ab. Die Buchstaben gestalten Sie, indem Sie mit dem Cursor in das Raster fahren. Wenn Sie die rechte Maustaste anklicken, löschen Sie einen Bildschirmpunkt, mit der linken Maustaste können Sie einen solchen Punkt setzen. Das weiße Feld links unten informiert Sie jeweils über das tatsächliche Aussehen des Buchstabens. Haben Sie Ihre Buchstaben fertig, klicken Sie »Schrift sichern« an. Wenn Sie dies vergessen, war Ihre Arbeit umsonst. Klicken Sie mit der rechten Maustaste »Ende« an und beantworten Sie die Frage, ob Sie den Fonteditor verlassen wollen, mit ja, indem Sie den Cursor auf »Ja« bewegen und mit der linken Maustaste anklicken. Nun laden Sie »Snapshot«.

## Mit Snapshot können Sie problemlos Grafiken in das Layout einbinden

Auf dem Bildschirm informiert Sie »Snapshot« darüber, daß Sie die Funktionen durch Betätigen von [Shift] und [PrtSc] aktivieren können. Jetzt ist es soweit – geben Sie auf Betriebssystemebene SDTP ein.

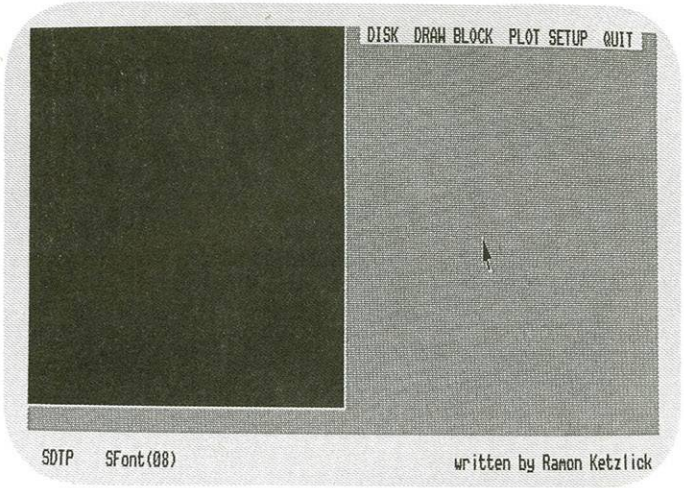



**Bild 3. So sieht SDTP nach dem Laden aus**

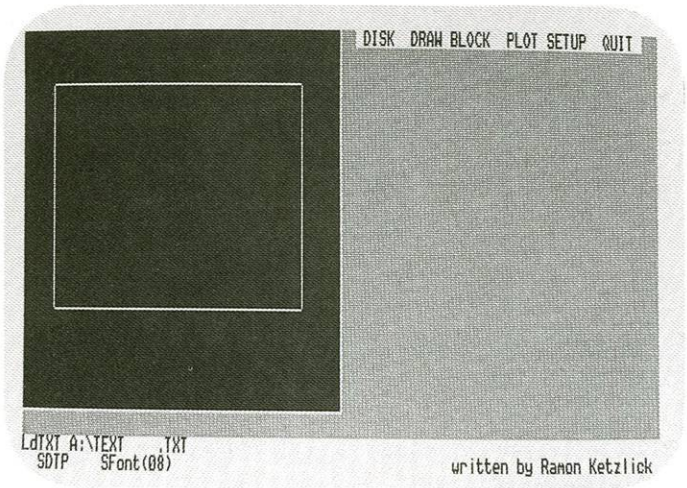



**Bild 4. In dieses Rechteck wird der Text gelegt**

Auf der linken Seite ist die Hauptmenüleiste sichtbar (Bild 3). Zunächst laden Sie einen beliebigen Text. Klicken Sie mit der linken Maustaste »Disk« an. Bewegen Sie nun den Mauscursor auf »ldtxt« und wiederholen sie den Klickvorgang. Der Text befindet sich in Laufwerk A. Klicken Sie mit der linken Maustaste in dem nun erschienenen Pop-Up-Menü die Raute mit den beiden Punkten an und dann – ebenfalls mit der linken Maustaste – das Laufwerk A. Das angemeldete Laufwerk ist nun das A-Laufwerk. Hier können Sie jetzt auch in Unterverzeichnisse wechseln. Wiederum mit der linken Maustaste wird nun das Feld angeklickt, in dem der Dateiname aufgeführt ist. Das Pop-Up-Menü verschwindet. Führen Sie den Mauscursor auf die weiße Fläche, zirka drei Zentimeter unter dem oberen Rand, und wiederholen Sie nochmals den Klickvorgang mit der linken Maustaste. Durch die Bewegung der Maus können Sie jetzt die Größe des Rechtecks bestimmen (Bild 4). Wenn Sie mit der Größe und Position zufrieden sind, betätigen Sie wieder die linke Maustaste, worauf ein Pop-Up-Menü erscheint, das Sie dazu auffordert, einzugeben, wie die Textdarstellung aussehen soll. Richten Sie den Cursor linksbündig aus und drücken Sie neuerlich die linke Maustaste. Das nächste Pop-Up-Menü erfragt von Ihnen, in wieviele Spalten der Text eingeteilt werden soll. Wählen Sie den Wert zwei und klicken Sie wiederum die linke Maustaste. Das Pop-Up-Menü verschwindet, das Laufwerk beginnt zu arbeiten und gibt Ihnen den Text zweispaltig in dem von Ihnen definierten Rechteck aus (Bild 5).

Auch nachträgliche Korrekturen sind kein Problem. Sind Sie der Meinung, Sie hätten den Text zu hoch plaziert, dann bewegen Sie den Cursor auf »Block«, klicken den Befehl an und wählen dann erneut mit der linken Maustaste den Befehl »set« an. Das Pulldown-Menü verschwindet, und Sie positionieren den Cursor an der linken oberen Ecke über dem Text und lassen die Maustaste los. So ergibt sich wieder ein Rechteck, das sie durch Betätigen der linken Maustaste gleichsam »festnageln«. Klicken Sie nun wieder den Befehl »Block« an und daraufhin mit der linken Maustaste den Befehl »Move«. Begeben Sie sich zurück zum Arbeitsbereich und bewegen Sie den Cursor mit gedrückter linker Maustaste nach unten. Das Rechteck erscheint wieder und sie positionieren neuerlich. Nach dem Drücken der linken Maustaste verschwindet der Cursor und der Text befindet sich dann um den von Ihnen bestimmten Abstand weiter nach unten versetzt. Gehen Sie nun mit dem Cursor wieder auf Disk und wählen Sie den Font wie bereits beschrieben an. Im Pop-Up-Menü entscheiden Sie sich jetzt für eine von Ihnen mit »Fonted« definierte – diesmal große Schrift – für die Überschrift. Gehen Sie nun zum Befehl »Draw« und wählen Sie im Untermenü »wrtxt«. Sie können nun in der vorletzten Zeile über die Tastatur einen beliebigen Text für die Überschrift eingeben. Nach dem Drücken der Taste [Return] wird links oben ein Rechteck dargestellt, das Sie mit der Maus beliebig positionieren können. Wenn Sie davon ausgehen, den richtigen Abstand erreicht zu haben, betätigen Sie wieder die linke Maustaste. Der Cursor

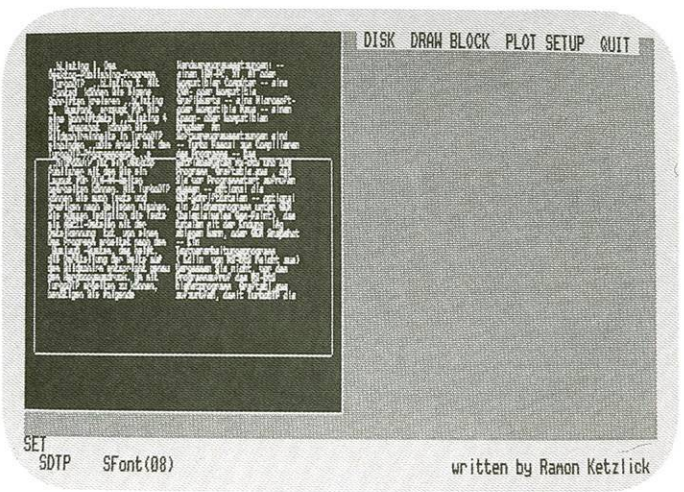



**Bild 5. Mit diesem Rechteck legen Sie die neue Position fest**

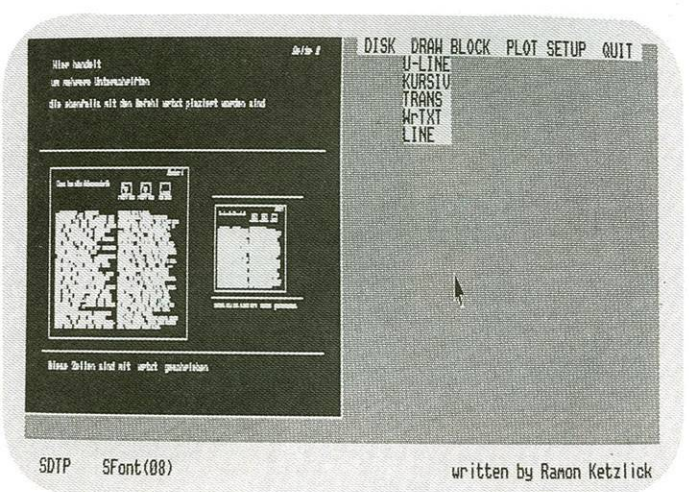



**Bild 6. Sie können die Überschrift auch direkt eingeben**

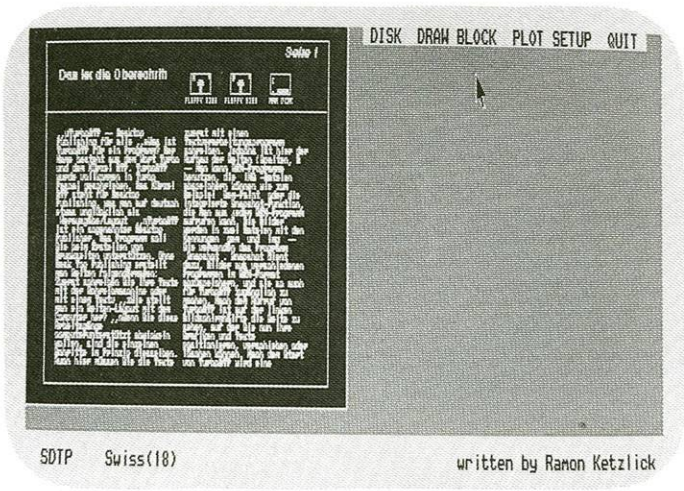



**Bild 7. Mit »line« können Sie Rahmen hinzufügen**

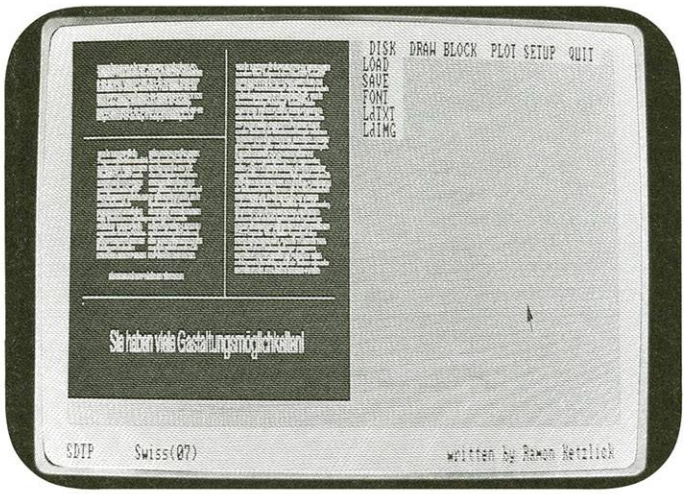



**Bild 8. Die Seitennummer, wieder direkt eingegeben, diesmal kursiv**

verschwindet, die Überschrift wird umgerechnet und erscheint zusammen mit dem Cursor (Bild 6).

Geben Sie die Befehle »Draw« und für das Untermenü »Line« an. Als nächstes bringen Sie den Cursor in der linken oberen Ecke der Arbeitsfläche in Position. Drücken Sie kurz die Maustaste, bewegen Sie den Cursor an das linke untere Ende, klicken Sie wieder kurz die Maustaste und führen Sie jetzt die Maus nach rechts. Sobald Sie davon ausgehen, den richtigen Abstand gefunden zu haben, führen Sie neuerlich den Klickvorgang mit der linken Maustaste aus, bewegen anschließend die Maus nach rechts oben und klicken nochmals an. Der Rahmen wird geschlossen, indem Sie mit der Maus wieder nach links fahren (Bild 7).

Ein kleiner Tip: Damit beim Schließen des Rahmens keine »schiefen Kanten« entstehen, empfiehlt es sich, beim Hochziehen der rechten Linie mit dem Cursor am Ausgangspunkt links oben aufzusetzen. Nach Beendigung dieses Vorgangs wird diesmal die rechte Maustaste betätigt – und der Rahmen ist fertig. Sie sollten es sich – wie auch bei Textverarbeitungsprogrammen – angewöhnen, von Zeit zu Zeit zwischenzuspeichern. Die einzelnen Erstellungsetappen sollten außerdem bezeichnende Namen erhalten. Dies erspart Ihnen viel Arbeit, wenn Sie beispielsweise das Gefühl haben, die Seite überladen zu haben. Greifen Sie nun wieder auf »Draw« zu, wählen Sie »kursiv« und dann, im gleichen Untermenü, »wrtxt« an. über die Tastatur geben Sie »Seite 1« ein und verfahren im übrigen wie bei der Überschrift (Bild 8).

## Sie können auf der Arbeitsfläche beliebig verschieben, löschen und kopieren

Nach Rückkehr zu »Draw« wird erneut »kursiv« angeklickt. Diese Schriftart ist damit wieder abgeschaltet. Sie können auch Grafiken in das Blatt einbinden, indem Sie bei Disk den Befehl »ldimg« anwählen (Bild 9).

Betätigen Sie jetzt einmal die Taste [Shift] und [Prtsc]. Damit haben Sie Snapshot aktiviert. Das Fadenkreuz wird in die linke obere Ecke bewegt und der Cursor mit gedrückter linker Maustaste zur rechten unteren Ecke geführt. Sobald Sie die Maustaste loslassen, speichert Snapshot den Inhalt im Rechteck. Die erzeugte Datei können Sie wiederum für ein Layout verwenden (Bild 10).

SDTP läuft zwar bereits mit nur einem Laufwerk; bezüglich der Ansprechzeiten müssen Sie sich bei großformatigen Texten und Grafiken allerdings in Geduld üben. Ansonsten sind Ihrer Kreativität mit SDTP keine Grenzen gesetzt. Sie können beispielsweise fertige Vorlagen – mit vordefinierten Rahmen und Grafiken – erstellen, in die Sie dann lediglich den jeweils neuen Text laden. Und wenn Ihnen noch neue Tricks einfallen – lassen Sie doch andere Leser auch daran teilhaben! (gh)

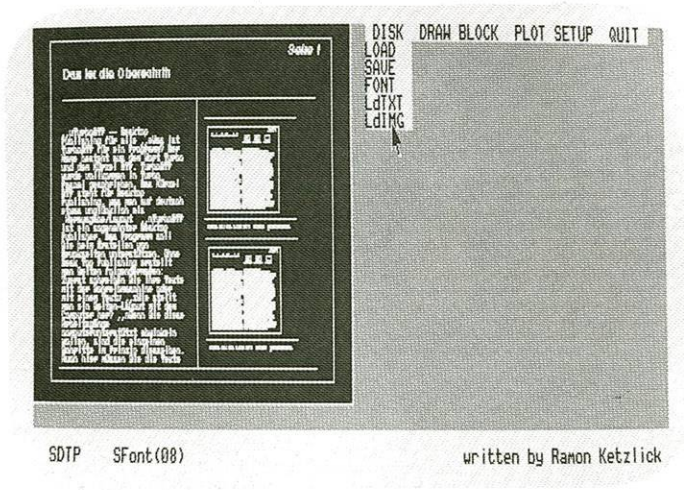



**Bild 9. Die Grafik, gespeichert mit Snapshot, kann man ohne Probleme einbinden**

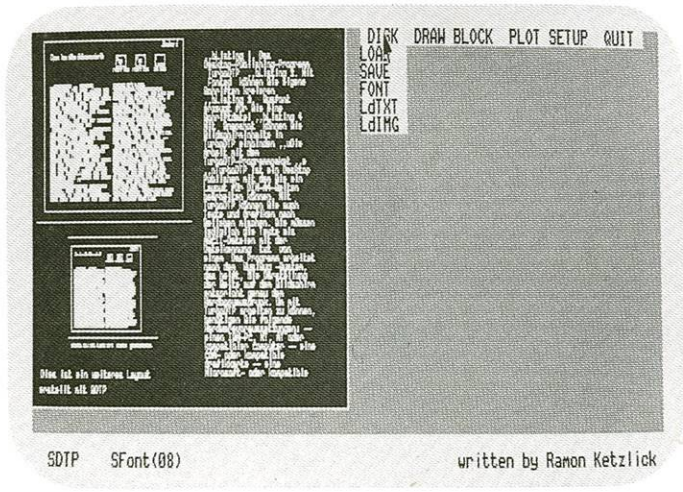



**Bild 10. Ein weiteres Beispiel für ein Seitenlayout**

# ConText PC

Autor: Matthias Uphoff

## So logisch und einfach kann Textverarbeitung sein!

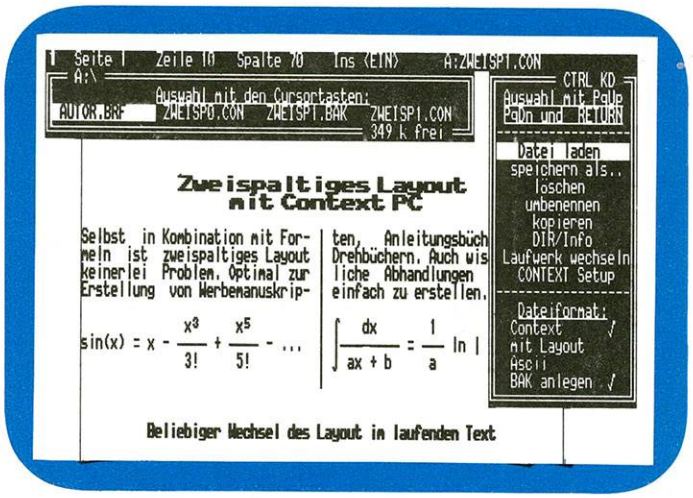
**Was Context PC nicht kann:**

- Dokumente mit 4000 Seiten verwalten
- 3-fach geschachtelte Fußnoten
- Grafik aus Fremdprogrammen einbinden.

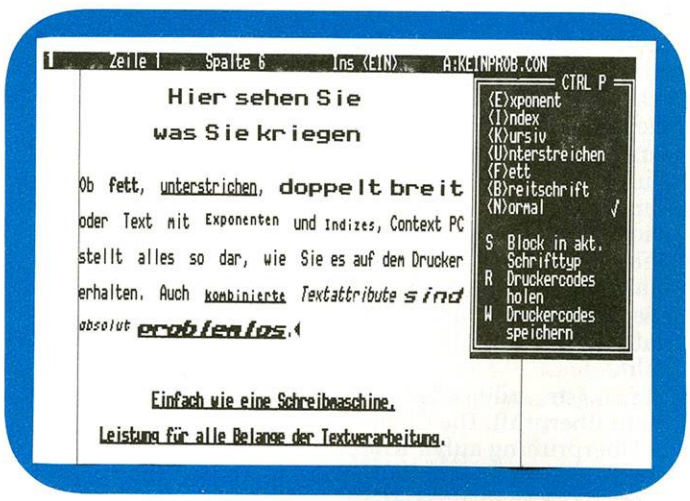
Wenn Sie diese Features unbedingt benötigen, empfehlen wir Ihnen die Textverarbeitung der Firma XXX!

**Was CONTEXT PC dafür jedoch ausgezeichnet beherrscht:**

- einfachste Bedienerlogik, Hilfsmenues für jede denkbare Situation
- exakte Wiedergabe des Druckbildes auf dem Bildschirm
- wahlweise eigene oder WordStar-kompatible Tastaturbelegung
- Textbausteine auf Tastendruck in Ihren Text einfügen
- Texte bis 65 Zeilen einfach mit einer Taste belegen
- perfekte Silbentrennung
- Rechnen im Text, automatische Datum- und Zeiteinfügung, Kalender
- Grafik-Zeichen direkt aus Tabelle in Text einsetzen
- Es können gleichzeitig 5 Texte unabhängig bearbeitet werden.
- Formatierung, die keine Wünsche offenläßt
- zweispaltiges Formatieren
- automatische BackUp-Funktion
- integrierter Dateimanager
- DOS-Aufrufe (auch andere Programme) aus ConText heraus starten
- durch hochspezialisierte Bildschirmtreiber auch im Graphikbetrieb außerordentlich schnell
- vielfältiger Einsatz in Batch-Dateien durch zahlreiche Möglichkeiten der Parameterübergabe
- hervorragend als Programm-Editor
- Programm im Hauptspeicher, keinerlei Diskettenzugriffe bei der Bearbeitung, ideal für Computer mit einem Laufwerk
- minimaler Speicherplatzbedarf bei maximaler Leistung durch äußerst effiziente Programmierung durch Assembler
- umschaltbare Textdarstellung während der Bearbeitung: 25/35 Zeilen (EGA + Hercules) 20/25 Zeilen (CGA)
- alle Standards einer Textverarbeitung: Suchen/Ersetzen auch mit Wild-Cards, Textmerker, jederzeit änderbare Randeinstellung, Blocksatz, Trennhilfe, 240 Zeichen Zeilenbreite, automatischer Tabulator, Zeile zentrieren
- arbeitet mit allen handelsüblichen Druckern
- umfangreiches Handbuch ca. 190 Seiten mit ausführlichen Erklärungen und Beispielen




*Der Dateimanager des Context PC macht DOS-Benutzeroberflächen überflüssig.*




*»WYSIWYG« in Reinform: Alle Schriftattribute werden am Bildschirm dargestellt.*

**Notwendige Konfiguration:**

PC XT/AT mit 256 kByte oder mehr, 1 Diskettenlaufwerk bzw. Festplatte, MS-DOS ab Version 2.0. CGA-, Hercules- und EGA-Grafikkarten werden unterstützt.
Zum Lieferumfang gehört ein umfangreiches deutsches Handbuch inklusive Tutorial.

© 1988 DMV GmbH

Best.-Nr. 228 5 1/4" Disk. Best.-Nr. 234 3 1/2" Disk.

# ConText PC

**99,– DM** (unverbindliche Preisempfehlung)

Wenn Sie über den DMV-Bestellservice bestellen, gilt folgendes:

| **Inland:** | | **Ausland:** | |
|---|---|---|---|
| Einzelpreis | 99,– DM | Einzelpreis | 99,– DM |
| zzgl. Versandkosten | 3,– DM | zzgl. Versandkosten | 5,– DM |
| **Endpreis** | **102,– DM** | **Endpreis** | **104,– DM** |

**– Bitte Bestellkarte benutzen –**

**DMV GmbH · Abt. Software · Postfach 250 · 3440 Eschwege**

# Tips & Tricks zu Pascal

**Ohne Übertreibung darf man wohl behaupten, daß Turbo Pascal zu den populärsten Programmiersprachen für Personalcomputer zählt. Mit Funktionen, Prozeduren, »Prögrämmchen« und Hinweisen sollen daher alle aktiven Pascal-Programmierer bei ihrer kreativen Tätigkeit unterstützt und davor bewahrt werden, immer wieder bei Punkt Null zu beginnen.**

## Editor statt Read-Befehl in Turbo Pascal

Tastatureingaben, die ein Programm verlangt, sind meistens einfache Kommandos und vor allem kurze Texte oder Zahlen. Wer die Editierfähigkeiten eines Textprogramms wie Wordstar kennt, vermißt einen ähnlichen Eingabekomfort bei dem Read-Befehl von Turbo Pascal.
Das Programm »ed_demo« benutzt die Include-Datei »ed.inc« (Listing 1). Diese enthält die Prozeduren »str_edit« und »tast«. »str_edit« stellt einige wichtige Editierfunktionen bereit. Folgende Parameter müssen beim Aufruf von »str_edit« angegeben werden:
»str« ist der zu editierende beziehungsweise neu einzugebende String.
»lmax« gibt die maximale Länge an, die »str« annehmen darf.
»col1« ist die Farbe, mit der das Editierfeld unterlegt wird.
»col2« ist die Farbe, die das Editierfeld bei Verlassen von »str_edit« annimmt.
»zahl_str« signalisiert, ob »str« eine Zahl repräsentiert; in diesem Fall gelten beim Editieren einige Besonderheiten.
»str_edit« zeigt an der laufenden Cursorposition beim Aufruf ein farbig unterlegtes Eingabefeld der Länge »lmax« und der Farbe »col1« mit je einem führenden und nachgestellten Leerzeichen an. Das Eingabefeld enthält den zu editierenden String oder ist leer, falls der String neu eingegeben werden soll.
Die Cursorposition und »lmax« müssen so gewählt sein, daß das Eingabefeld nicht über die laufende Zeile hinausgeht.
Die an »str_edit« übergebenen Längen der Strings werden nicht überprüft. Die Compileranweisung »$V – « setzt diese Überprüfung außer Kraft. »$V – « ist aber nur im Hauptprogramm und nach der Deklaration aller beteiligten Strings wirksam.
Der Editor führt folgende Cursorbewegungen aus:
[Ctrl-D], [Cursor-rechts] – ein Zeichen nach rechts;
[Ctrl-F], [Ctrl-H], [Cursor-links] – ein Zeichen nach links;
[Home] – zum Anfang des Feldes;
[End] – zum Ende des Strings.

Es gibt zwei Löschfunktionen:
[Ctrl-G], [Del-Cursor-rechts] – Zeichen löschen;
[Ctrl-T] – alle Zeichen rechts des Cursors löschen.
[Return], [Cursor-aufwärts], [Cursor-abwärts] und [Ctrl-F10] beenden jeweils die Prozedur.

Da die Case-Anweisung erlaubt, Marken in Gruppen anzugeben, ist es einfach, beliebige andere Tastencodes ganz nach Wunsch vorzugeben.

»str_edit« setzt die Prozedur »tast« voraus, die die Tastatur liest und auch die Tastencodes bildet. »tast« wandelt die Tastaturzeichen in deren Ordnungszahlen um. Handelt es sich bei der Eingabe um Funktionstasten- oder andere Sondereingaben, erhalten die Ordungszahlen ein negatives Vorzeichen. Dadurch werden alle Tastatureingaben zu Integerzahlen, die von der Case-Anweisung einheitlich zu behandeln sind. Nebenbei sei bemerkt, daß sich »tast« auch unabhängig von »str_edit« aufrufen läßt.

Beide Prozeduren und auch das Hauptprogramm (Listing 2) benutzen die globale Variable »key«, die den Tastaturcode enthält; sie ist in der Datei »ed.inc« deklariert.

»str_edit« darf mit mehreren Tastencodes beendet werden, die jeweils in »key« enthalten sind. Daher ist mit dieser Variablen die Fortsetzung des Hauptprogramms zu steuern, ohne daß dazu eine besondere Eingabe notwendig wäre. Das Hauptprogramm beispielsweise testet »key«, um gegebenenfalls das Programm zu beenden.

Beim Editieren werden darstellbare Zeichen in den String eingefügt. Füllt der String das Eingabefeld ganz aus, werden die Zeichen an der Cursorposition überschrieben.

Editiert man einen String aus Ziffern, so geschieht dies im Modus »Überschreiben«. Das wird dem Editieren von Zahlen besser gerecht als das Einsetzen von Zeichen. Außerdem werden keine Leerzeichen angenommen – weder direkt noch indirekt –, da bei der Umwandlung von Zahlenstrings Leerzeichen zu Fehlern führen. Der Cursor kann deshalb den Zahlstring nicht verlassen.

Zahlen mit Hilfe eines Strings einzugeben, hat unter anderem den Vorteil, daß keine Ein- und Ausgabe-Fehler entstehen, die abzufangen sind. Der String muß allerdings mit der Val-Prozedur in eine Zahl umgewandelt werden, wie dies das Hauptprogramm zeigt. Entsteht hierbei ein Fehler, ruft man einfach »str_edit« bis zur Fehlerfreiheit immer wieder auf. Die Val-Prozedur erwartet einen Zahlenstring, der nach den Regeln von Turbo Pascal, dem Typ von »zahl« entsprechend, aufgebaut ist. Beim Verlassen von »str_edit« wird die Hintergrundfarbe »col1« durch die Farbe »col2« ersetzt.

(Dieter Holzhäuser/zi)

```
[000]    0:
[HWE]    1: { ed.inc,   Tastaturabfrage und Stringedito
           r }
[000]    2:
[UM5]    3: var key : integer;
[000]    4:
[56D]    5: procedure tast;                {Tastatur les
           en}
[6C4]    6:  var   ta : char;
[BH1]    7: begin
[RH3]    8:   read (kbd,ta);
[LL3]    9:   key := ord (ta);
[1YB]   10:   if (ta = chr(27)) and keypressed then beg
           in
[GY3]   11:     read (kbd, ta);
[P64]   12:     key := - ord(ta);
[S0A]   13:   end;                    {Normaltaste pos. Za
           hl}
```

```
[NVA]   14: end;                      {Sondertaste neg. Za
           hl}
[000]   15:
[D95]   16:  {Stringeditor}
[AV7]   17: type   strtyp75 = string[75];
[000]   18:
[706]   19: procedure str_edit
[UP8]   20:   ( var  str : strtyp75;  {String}
[6DA]   21:     lmax,                 {Max. Stringlänge
                }
[YDA]   22:     col1,                 {Feldhintergr.far
           be   }
[UNB]   23:     col2     : byte;      { dto. bei Verlas
           sen  }
[KVD]   24:     zahl_str : boolean ); {Flagge zeigt Zah
             lstr.}
[000]   25:
[V1A]   26: var  pos,                 {rel. Cursorpos.}
```

```
[UE6]   27:      e,                          {Zähler          }
[C99]   28:      xorg  : byte;               {Feldbeginn      }
[4FA]   29:      immer : boolean;            {Ende-Flagge     }
[000]   30:
[HK6]   31: procedure curs_left;
[BH1]   32: begin
[YYC]   33:   if pos > 1 then begin   {Cursor nach link
          s}
[9E4]   34:     pos := pred(pos);
[YE7]   35:     gotoxy (xorg + pos ,wherey);
[TU0]   36:   end;
[TU0]   37: end;
[000]   38:
[MRE]   39: procedure curs_right;     {Cursor nach rech
          ts}
[BH1]   40: begin
[4C7]   41:   if pos < lmax then begin
[758]   42:     gotoxy (wherex+1,wherey);
[5E4]   43:     pos := succ(pos);
[TU0]   44:   end;
[TU0]   45: end;
[000]   46:
[BH1]   47: begin
[XM6]   48:   textbackground (col1);
[000]   49:
[BDF]   50:   str := copy (str,1,lmax); {str auf lmax v
          erkürzen}
[LM4]   51:   xorg := wherex;
[484]   52:   write (' ',str);
[53C]   53:   for e := length(str) to lmax do write(' '
          );
[97E]   54:     {falls str kürzer lmax Leerzeich. ausge
          ben }
[0E6]   55:   gotoxy(xorg+1, wherey);
[KP1]   56:   pos := 1;
[V14]   57:   immer := true;
[X52]   58:   repeat
[1L1]   59:     tast ;
[713]   60:     case key of
[6T4]   61:       { ^D, Right-arrow}
[7VF]   62:       { falls zahl_str kann cursor string n
          icht verl.}
[U88]   63:       4,-77: if zahl_str then begin
[HBC]   64:                if pos <= length(str) then
          curs_right;
[PD6]   65:              end else curs_right;
[000]   66:
[908]   67:      19,-75,                {^S, BS, Left-a
          rrow}
[H04]   68:           8: curs_left;
[000]   69:
[DX5]   70:       7,-83: begin           {^G, Del}
[CG5]   71:                delete (str,pos,1);
[LTC]   72:                write (copy(str,pos,(length(
          str)-pos+1)),' ');
[TK7]   73:                gotoxy(xorg + pos, wherey);
[TU0]   74:              end;
[000]   75:
[WF4]   76:       -71  :                 {Home}
[KE7]   77:              while pos > 1 do curs_left;
```

```
[000]   78:
[3Z3]   79:       -79  :                 {End}
[DU7]   80:              while (pos < length(str)) or
[NL6]   81:                    ((pos = length(str)) and
[Y2A]   82:                     (length(str) < lmax)) d
          o curs_right;
[000]   83:
[BV3]   84:         20 :                 {^T}
[TV9]   85:              if pos <= length(str) then beg
          in
[ALA]   86:                delete (str,pos,length(str)-
          pos+1);
[53C]   87:                for e := length(str) to lmax
           do write(' ');
[KE7]   88:                gotoxy (xorg + pos, wherey);
[TU0]   89:              end;
[000]   90:
[ZH4]   91:                                    {Ret, Down- an
          d  }
[WL8]   92:       13,-72,-80,                  {Up-arrow, Ctr
          l F10}
[Z15]   93:       -103 : immer := false;
[000]   94:
[36E]   95:       else {druckbare Zeichen  eins./anh./ü
          berschr.
[5SD]   96:            falls zahl_str, wird leer nicht
           angenommen
[Y3C]   97:            und es ist nur Überschreiben mö
          glich }
[000]   98:
[029]   99:         if ((key > 31) and not zahl_str) or
[AC9]  100:            (zahl_str and (key > 32)) then
          begin
[000]  101:
[K1B]  102:           while (length(str)+1) < pos do st
          r := str + ' ';
[4F7]  103:           if (length(str) = lmax) or
[5V9]  104:              zahl_str then delete(str,pos,1
          );
[B28]  105:           insert (chr(key),str,pos);
[DLB]  106:           write (copy(str,pos,length(str)-p
          os+1));
[PZ9]  107:           if pos < lmax then pos := succ(po
          s);
[TK7]  108:           gotoxy(xorg + pos, wherey);
[TU0]  109:         end;
[000]  110:
[8T5]  111:     end;                       {case end}
[000]  112:
[NS4]  113:   until  not immer;
[000]  114:
[VDF]  115:   {Feld normal anzeigen, da str_edit verlas
          sen wird}
[000]  116:
[1N6]  117:   textbackground (col2);
[3D6]  118:   gotoxy (xorg,wherey);
[484]  119:   write (' ',str);
[53C]  120:   for e := length(str) to lmax do write(' '
          );
[TU0]  121: end;
```

**Listing 1. Diese Include-Datei macht den Stringeditor nutzbar**

```
[2RF]    0: program ed_demo;       {String-Editor-Demons
           tration}
[H51]    1: {$C-}
[000]    2:
[VM1]    3: const
[1DE]    4:    feldtext1 : string[25] =  'Das ist ein T
           esttext';
[YL8]    5:    feldtext2 : string[17] =  '12.34';
[000]    6:
[EM0]    7: var
[9T4]    8:   code       :integer;
[6W3]    9:   zahl       :real;
[000]   10:
[TT2]   11: {$I ed.inc }
[LPH]   12:   {die Integer-Variable key ist in ed.inc g
           lobal deklariert}
[000]   13:
[ENE]   14: {$V-}                 {Stringlängenprüfung ab
           schalten}
[000]   15:
[HCA]   16: begin                                  {Haupt
           programm}
[0M8]   17:   textbackground (lightgray);
[S26]   18:   textcolor      (black);
[NM2]   19:   clrscr;
[ZK1]   20:   write
[M69]   21:   ('              EDITIEREN IN FELDERN
           -      DEMO');
[1B3]   22:   gotoxy (1,6);
[0KC]   23:   write  ('Beispiel für einen Textstring:')
           ;
```

```
[PL3]   24:   gotoxy (1,10);
[YYC]   25:   write  ('Editieren eines Zahlenstrings:')
           ;
[DM3]   26:   gotoxy (1,25);
[R45]   27:   write ('NEXT: ',#24,' ',#25,
[F8A]   28:          ' Ret ▌ END: Ctrl F10 ▌ Cursor: ',
[513]   29:          #26,#32'^D '#27,#32,
[529]   30:          '^S Home End ▌ Lösch.: ^G  Del ^T'
           );
[6M3]   31:   gotoxy (1,14);
[PSB]   32:   write  ('Umgewandelter Zahlenstring :');
[000]   33:
[X52]   34:   repeat
[LM3]   35:     gotoxy (35,6);
[LR9]   36:     str_edit (feldtext1, 25, 5, 7, false);
[RP6]   37:     if key = -103 then exit;
[7Z3]   38:     gotoxy (35,14);
[NL2]   39:     clreol;
[X52]   40:     repeat
[LY3]   41:       gotoxy (35,10);
[ZE9]   42:       str_edit (feldtext2, 17, 3, 7, true )
           ;
[RP6]   43:       if key = -103 then exit;
[M77]   44:       val (feldtext2, zahl, code);
[1Y3]   45:     until code = 0;
[7Z3]   46:     gotoxy (35,14);
[KK3]   47:     write  (zahl);
[4B3]   48:   until false;
[5T0]   49: end.
```

**Listing 2. Das Programm veranschaulicht die Fähigkeiten des Stringeditors**

# Tips & Tricks zu Basic

**Besonders Basic-Programmierer sind ständig auf der Suche nach Tricks. Im folgenden finden Sie eine Auswahl von Peeks, Pokes und anderen Kniffen. Ein Schwerpunkt dabei ist die Programmierung der Laufwerke von Basic aus. Anhand der gezeigten Tricks ist es kein Problem, ein kleines Diskettendiagnose-Programm selbst zu schreiben.**

## Feststellen des Computertyps

Es gibt einen einfachen Trick, um festzustellen, welcher Computer gerade benutzt wird. Die Adresse 65534 ist für ein »Computertyp-Byte« reserviert, wobei gilt:

```
255 - Original IBM-PC
254 - IBM XT oder IBM Portable
253 - IBM PC Junior
252 - IBM AT
andere - kompatibler Computer
```

Dieses Byte läßt sich beispielsweise in Basic so ermitteln:

```
def seg = &hf000
print "Das Byte des Computertyps ist: ";peek (&hfffe)
```

(Benny Kleinke/tr)

## Abfrage der Sondertasten

Um festzustellen, ob und welche Sondertasten (»Insert«, »Caps-Lock«, und so weiter) gedrückt sind, kann man das Byte an der Adresse 1047 abfragen. Die Konfigurationen aus der Tabelle sind möglich. Diese lassen sich beispielsweise mit folgender Routine ermitteln:

```
10 rem gedrückte Tasten anzeigen
20 def seg=0
30 a=peek(1047)
40 rem
50 ins=a and 2↑7
60 cap=a and 2↑6
70 num=a and 2↑5
80 scr=a and 2↑4
90 a_s=a and 2↑3
100 c_s=a and 2↑2
110 shl=a and 2↑1
120 shr=a and 2↑0
130 rem
140 if ins=2↑7 then print "Insert-Modus aktiv"
150 else print "Insert-Modus nicht aktiv"
160 if cap=2↑6 then print "Caps-Lock-Modus aktiv"
170 else print "Caps-Lock-Modus nicht aktiv"
180 if num=2↑5 then print "Num-Lock-Modus aktiv"
190 else print "Num-Lock-Modus nicht aktiv"
200 if scr=2↑4 then print "Scroll-Lock-Modus aktiv"
210 else print "Scroll-Lock-Modus nicht aktiv"
220 if a_s=2↑3 then print "Alt-und-Shift-Modus aktiv"
230 else print "Alt-und-Shift-Modus nicht aktiv"
240 if c_s=2↑2 then print "Ctrl-und-Shift-Modus aktiv"
230 else print "ctrl & shift Modus nicht aktiv"
260 if shl=2↑1 then print "Shift-links-Modus aktiv"
270 else print "Ctrl-und-Shift-Modus nicht aktiv"
280 if SHR=2↑0 then print "Shift-rechts-Modus aktiv"
290 else print "Shift-rechts-Modus nicht aktiv"
300 end
```

(Benny Kleinke/tr)

## Feststellen, ob der Diskettenmotor läuft

Die Speicheradresse 1087 enthält den Diskettenmotor-Status. Er gibt an, bei welchem Laufwerk gerade der Motor läuft und welches Laufwerk gerade adressiert wird. Folgende Kombinationen sind möglich:

| peek(1087) and 7 | Bedeutung |
|---|---|
| 0 | Kein Motor aktiv |
| 1 | Laufwerk A wird adressiert |
| 2 | Laufwerk B wird adressiert |
| 3 | Laufwerk C wird adressiert |
| 4 | Laufwerk D wird adressiert |

Abfragen lassen sich diese Werte mit folgender Basic-Zeile:

```
def seg=0 : print peek(1087) and 7
```

Diese Adresse ist vor allem dann nützlich, wenn zum Beispiel ein Programm »hängt«, weil ein Diskettenzugriff nicht ordnungsgemäß beendet wurde.

(Benny Kleinke/tr)

## Die momentane Position des Schreib-/Lesekopfes

Interessiert man sich bei einer defekten Diskette dafür, welche Spur und welcher Sektor für diesen Defekt verantwortlich sind, empfiehlt sich ein Blick auf die Positionswerte des Schreib-/Lesekopfes. Sie sind in den Adressen 1093 bis 1096 gespeichert. In Adresse 1093 ist die Nummer der aktuellen Spur abgelegt, Adresse 1094 gibt Auskunft über die Anzahl der Köpfe, und in der Adresse 1095 steht, welcher Sektor gerade gelesen wird.

Die Adresse 1096 hat eine besondere Funktion: Sie dient dazu, die Anzahl der Bytes des aktuellen Sektors zu speichern. Diese Byte-Anzahl ist allerdings nicht direkt abgelegt, sondern sie muß aus dem Inhalt von 1096 errechnet werden. Das geschieht folgendermaßen:

```
def seg=0: print 128*2↑peek(1096)
```

Um nun den aktuellen Sektor und die aktuelle Spur herauszufinden, kann man dieses Demonstrationsprogramm verwenden:

```
10 def seg=0
20 spu=peek(1093)
30 sek=peek(1095)
40 kop=peek(1094)
50 byt=128*2↑peek(1096)
60 print "Der Schreib/Lesekopf ";kop" befindet sich auf Spur
";spu" im Sektor ";sek
70 print "Die Anzahl Bytes in diesem Sektor beträgt ";byt
```

Zusammen mit dem weiter oben beschriebenen Trick zum Herausfinden des aktuellen Laufwerks läßt sich so ein kleines Diskettendiagnose-Programm zurechtbasteln.

(Benny Kleinke/tr)

| Bit | Bedeutung bei gesetztem Bit |
|---|---|
| 7 | Insert-Modus aktiv |
| 6 | Caps-Lock aktiv |
| 5 | Num-Lock aktiv |
| 4 | Scroll-Lock aktiv |
| 3 | [Alt] und [Shift] betätigt |
| 2 | [Ctrl] [Shift] betätigt |
| 1 | [Shift-links] betätigt |
| 0 | [Shift-rechts] betätigt |

**Tabelle. Über das Byte an der Adresse 1047 lassen sich die verschiedenen Konfigurationen abfragen**

## Diskette auf Schreibschutz überprüfen

Versucht man von einem Basic-Programm aus schreibend auf eine Diskette zuzugreifen, die mit einem Schreibschutz versehen ist, wird entweder ein On-error-goto-Sprung ausgelöst, oder im schlimmeren Fall das Programm beendet. Um ein Programm gegen solche »Unfälle« abzusichern, sollte es zunächst überprüfen, ob die im Laufwerk befindliche Diskette schreibgeschützt ist. Hierfür benötigt man lediglich den Inhalt von Adresse 1090.

```
10 def seg=0
20 a=(peek(1090) and 3)
30 if a=3 then print "Die Diskette ist schreibgeschützt!" : end
```

Vor der Anwendung dieser Routine muß ein Diskettenzugriff erfolgt sein, damit sichergestellt ist, daß in Adresse 1090 der richtige Wert steht. (Benny Kleinke/tr)

## Zahl der angeschlossenen Laufwerke

Durch Auslesen der Adresse 1040 läßt sich feststellen, ob überhaupt Laufwerke an den Computer angeschlossen sind, und wenn ja, wieviele. Hierbei werden aber nicht die Bezeichnungen der einzelnen Laufwerke ermittelt, sondern nur der entsprechende Zahlenwert. Durch Start der Routine

```
10 def seg=0
20 z=peek(1040) and 1
30 az=peek(1040) and 192
40 if z=0 then print "Es ist kein Laufwerk vorhanden."
50 if az<>0 then print "Die Zahl der vorhandenen Laufwerke beträgt ";az/64+1
```

werden diese Werte ausgegeben und für eine spätere Auswertung bereitgestellt. (Benny Kleinke/tr)

## Größe des RAM

Um herauszufinden, wieviel Hauptspeicherplatz in dem jeweiligen Rechner vorhanden ist, untersucht man den Inhalt der Adressen 1043 und 1044. Diese Adressen enthalten die genaue Speichergröße des PC in KByte in der Adressierungsart »Wort«.

```
10 def seg=0
20 print "Die RAM-Größe Ihres PCs beträgt
";peek(1043)+256*peek(1044);" K"
```

An dieser Stelle sollte nebenbei erwähnt werden, daß das »K« hinter Byte-Größen nicht die Abkürzung für »Kilo« ist, wie sehr häufig angenommen wird. Es steht vielmehr für die Größe »1024 Bytes«. Also:
1 kilo-Byte = 1000 Byte, aber
1 KByte = 1024 Byte (Benny Kleinke/tr)

## Softwaregesteuertes Setzen der Tastatur-Modi

Durch einen Poke-Befehl läßt sich beispielsweise die Taste [Caps-Lock] aktivieren. Dabei wird das entsprechende Bit für »an« gesetzt und das Bit für »aus« gelöscht. Am Anfang eines solchen Programms muß stehen:

```
def seg=0
a=peek(1047)
```

Dann können über die nachfolgenden Zeilen die verschiedenen Modi aktiviert beziehungsweise ausgeschaltet werden:

```
Insert on: poke 1047,a or 128
Insert off: poke 1047,a or 127
Caps-Lock on: poke 1047,a or 64
Caps-Lock off: poke 1047,a or 191
Num-Lock on: poke 1047,a or 32
Num-Lock off: poke 1047,a or 223
Scroll-Lock on: poke 1047,a or 16
Scroll-Lock off: poke 1047,a or 239
```

(Benny Kleinke/tr)

## Sperren der Tastatur

Hin und wieder ist es von Nutzen, die Tastatur gegen Eingaben des Anwenders sperren zu können. Dies ist durch eine einzelne Poke-Anweisung zu bewerkstelligen:

```
def seg=0
poke 97,204
```

Entriegeln läßt sich die Tastatur mit

```
poke 97,76
```

Achtung: Diese Befehle sollten nicht im Direktmodus angewendet werden, da die Tastatur sonst den Befehl zum Entriegeln nicht mehr annehmen kann. Holzauge sei wachsam...

(Benny Kleinke/tr)

## Tastaturpuffer löschen

Werden während des Ablaufs eines Programms Tasten betätigt, so werden sie im Tastaturpuffer gespeichert. Bei der nächsten Input-Anweisung werden diese Zeichen als Eingabe ausgewertet und erscheinen in der Eingabezeile. Beinhaltet jedoch das Programm statt der Input-Anweisung eine Inkey$-Abfrage, dann erweist sich dieser Umstand der zwischengespeicherten Zeichen im Tastaturpuffer als ziemlich unangenehm. In diesem Fall kann es vorkommen, daß ungewollt Optionen ausgeführt werden.

Deswegen ist es sinnvoll, vor einer Eingabeanweisung den Tastaturpuffer zu löschen. Das kann durch folgende Anweisungen geschehen:

```
10 def seg=0
20 poke 1050,peek(1050)
30 poke 1051,0
```

Wer bedienungssichere Programme schreiben möchte, sollte von dieser Methode häufig Gebrauch machen.

(Benny Kleinke/tr)

## Umgehen des Listschutzes beim Basic-Interpreter

Wurde ein Programm mit »save "Titel",p« gespeichert, ist es nicht mehr mit »list« einzusehen. Wenn man nur noch über eine so geschützte Version des Programms verfügt und trotzdem Modifikationen daran vornehmen möchte, wendet man folgenden Trick an:

```
load "Titel"
def seg:
new:
bsave "re_list",1024,300
```

Auf diese Weise wird eine Datei »re_list« erzeugt, die Programme in dem Zustand enthält, in dem sie noch nicht mit einem List-Schutz versehen sind. Nun kann nach dem Laden des geschützten Programms mit der Anweisung »bload "re_list",1024« der List-Schutz entfernt werden.

(Benny Kleinke/tr)

## Unterdrücken von »Break« und »Ctrl-C«

Um zu verhindern, daß Unbefugte ein Programm abbrechen, läßt sich mit folgender Routine die Break-Taste und die Tastenkombination [Ctrl-C] sperren:

```
10 def seg=0
20 for i=&h6c TO &h6f
30 read p
40 poke i,p
50 next i
60 data 83,255,0,240
```

(Benny Kleinke/tr)

## Softwaregesteuerte Hardcopy

Diese Routine erlaubt es, von einem Programm aus eine Hardcopy herzustellen. Es ist dabei wichtig, ebenso wie bei der manuell aktivierten Hardcopy-Funktion, sich rechtzeitig zu vergewissern, daß der Drucker eingeschaltet und druckbereit ist.

```
10 for i=1 to 3
20 read p
30 a$=a$+chr$(p)
40 next I
50 data 205,5,203
60 hcopy = varptr(a$)
```

Der Aufruf der Hardcopy-Routine erfolgt dann mit »call hcopy«.

(Benny Kleinke/tr)

# Tips und Tricks für C-Programmierer

## Gesetzte Bits feststellen

Häufig trifft man in Programmbeschreibungen auf den Hinweis, daß bestimmte Bits von spezifizierten Adressen über vorhandene oder nicht vorhandene Eigenschaften des benutzen Computers Auskunft geben. Wenn Sie wissen wollen, ob ein PC diese Eigenschaften besitzt, müssen Sie nur die zwei kleinen Programme (Listing 1 und 2) abtippen. Nach der Eingabe der Segment- und Offset-Adresse lesen die Programme »liesbyte.c« und »liesbits.c« sofort den entsprechenden Bereich beziehungsweise das entsprechende Byte aus und geben den Inhalt auf den Bildschirm aus.

Die Ausgabe von »liesbits.com« erfolgt mit einer Schablone von 15 bis 0, so daß Sie direkt ablösen können, welche der Bits gesetzt sind (Ausgabe »ff«).

Das Programm »liesbyte.com« liest den gesamten angegebenen Adreßbereich aus, unterdrückt aber vor der Bildschirmausgabe die Kontrollzeichen sowie die ASCII-Werte ab 7Fhex. Dadurch vermeidet man den durch die Kontrollzeichen hervorgerufenen, unkontrollierten Abbruch. Sie können natürlich den Ausgabefilter Ihren individuellen Bedürfnissen anpassen, indem Sie die Werte 0x20 beziehungsweise 0x7F in der Abfrage »if (byte ...)« ändern. Das Programm leistet auch bei der Fehlersuche gute Dienste. Sie könnten es beispielsweise einsetzen, um bestimmte Kontrollzeichen zu suchen. Sie müssen die Programme mit der Compiler-Einstellung »tiny« compilieren und mit »exe2bin« in eine Com-Datei umwandeln.

(Jürgen Hochstädt/gh)

## Turbo C und die Compiler-Optimierungseinstellungen

Stehen Sie manchmal auch vor der Entscheidung: Geschwindigkeit oder Größe des vom Compiler zu erzeugenden Codes optimieren zu müssen? Vielleicht sind folgende Auswertungen eine kleine Hilfe für Sie (Tabelle 1). Als Testgerät wurde ein 8 MHz Turbo XT mit den Mikroprozessor NEC V 20 verwendet. Es wurde mit der Turbo C Version 1.0 gearbeitet. Die Datei »programm.c« hatte eine Größe von 19410 Byte – 1216 Zeilen ohne Linkerzusatz – und ist nicht auf bestimmte Funktionen spezialisiert. Eine Meßstrecke der Laufzeit eines Teiles des ablauffähigen Programms wurde eingebaut.

In Anbetracht der Tatsache, daß selbst das kleinste Programm von nur 10 Byte Länge auf einer Festplatte einen Cluster mit 4 Sektoren, also 2048 Byte belegt, ist also eine Optimierung der Laufzeit in Verbindung mit Sprungbefehlen und Registern zu empfehlen. Lediglich bei Computern ohne Festplatte lohnt es sich schon, eine Berechnung anzustellen, da die Minimalbelegung der Diskette nur 1024 Byte beträgt.

Natürlich können Programme, die beispielsweise umfangreiche Datenbewegungen beinhalten, ganz andere Werte ergeben. Vielleicht haben Sie bereits Erfahrungen gesammelt und könnten dazu einen Tip beisteuern?

(Jürgen Hochstädt/gh)

## Verwendung von »printf«, »write(str)« und »puts(putch)«

Wie Sie sicher wissen, stellt Turbo C in den entsprechenden Header-Dateien verschiedene Funktionen für die Bildschirmausgabe bereit. Um diese Funktionen auf ihr Laufzeitverhalten zu testen, entstand ein Programm, daß 2000mal den Buchstaben »A« auf den Bildschirm ausgibt. Ein zweites Programm brachte 50mal einen 40 Zeichen langen String auf den Bildschirm. Auf der Hardwareseite

```
[C85]    0: #include <dos.h>
[TV5]    1: #include <stdio.h>
[356]    2: #include <cursor.h>
[000]    3:
[Z74]    4: long oa,oe,sg;
[000]    5:
[0P1]    6: main() {
[000]    7:
[MT2]    8: char byte;
[UM1]    9: int k;
[M11]   10: k = 0;
[000]   11:
[BK2]   12: clrscr();
[P43]   13: gotoxy(0,0);
[000]   14:
[8C2]   15: while(1) {
[91D]   16: printf("\n\n Segment ..... : ");  cscanf("%
            l4X",&sg);
[FXC]   17: printf("\n Anfang Offset : ");  cscanf("%l4
            X",&oa);
[CWC]   18: printf("\n Ende   Offset : ");  cscanf("%l4
            X",&oe);
[GB3]   19: printf("\n");
[206]   20:   for(k=oa; k<oe; k++) {
[3R4]   21:    byte = peekb(sg,k);
[7PC]   22:     if(byte < 0x20)  { write('*',121); cont
            inue; }
[61D]   23:     if(byte > 0x7f)  { write('*',121); cont
            inue; }
[W78]   24:    printf("%c",byte);  }  /* for */
[LV2]   25:          } /*while */
[CV2]   26: } /* main() */
```

**Listing 1. Das Programm »liesbyte.c«**

```
[C85]    0: #include <dos.h>
[TV5]    1: #include <stdio.h>
[356]    2: #include <cursor.h>
[000]    3:
[303]    4: long oa,sg;
[000]    5:
[0P1]    6: main() {
[000]    7:
[MT2]    8: char byte;
[XGG]    9: unsigned char b15,b14,b13,b12,b11,b10,b9,b8
            ,b7,b6,b5,b4,b3,b2,b1,b0;
[000]   10:
[BK2]   11: clrscr();
[P43]   12: gotoxy(0,0);
[000]   13:
[8C2]   14: while(1) {
[RCC]   15: printf("\n\n Segment : ");  cscanf("%l4X",&
            sg);
[MFB]   16: printf("\n Offset  : ");  cscanf("%l4X",&oa
            );
[8A4]   17: printf("\n\n");
[D05]   18:    byte = peekb(sg,oa);
[VY5]   19:    printf(" %c \n",byte);
[000]   20:
[Y53]   21:    b15 = (byte >> 15);
[N44]   22:    b14 = (byte << 1 >> 15);
[K44]   23:    b13 = (byte << 2 >> 15);
[F44]   24:    b12 = (byte << 3 >> 15);
[C44]   25:    b11 = (byte << 4 >> 15);
[944]   26:    b10 = (byte << 5 >> 15);
[ME4]   27:    b9 = (byte << 6 >> 15);
[TE4]   28:    b8 = (byte << 7 >> 15);
[YE4]   29:    b7 = (byte << 8 >> 15);
[3F4]   30:    b6 = (byte << 9 >> 15);
[RA4]   31:    b5 = (byte << 10 >> 15);
[NA4]   32:    b4 = (byte << 11 >> 15);
[LA4]   33:    b3 = (byte << 12 >> 15);
[HA4]   34:    b2 = (byte << 13 >> 15);
[FA4]   35:    b1 = (byte << 14 >> 15);
[DA4]   36:    b0 = (byte << 15 >> 15);
[000]   37:
[AKC]   38:    printf(" Bit: 15 14 13 12 11 10 09 08 07
            06 05 04 03 02 01 00 \n\n");
[TRF]   39:    printf("     %3x%3x%3x%3x%3x%3x%3x%3x%3x
            %3x%3x%3x%3x%3x%3x%3x \n\n",
[MSB]   40:                  b15,b14,b13,b12,b11,b10,b9,
            b8,b7,b6,b5,b4,b3,b2,b1,b0);
[000]   41:
[LV2]   42:          } /*while */
[CV2]   43: } /* main() */
```

**Listing 2. Das dazugehörende Programm »liesbit.c«**

fand ein IBM Turbo XT Vewendung, der mit 8 MHz taktet und als Prozessor den NEC V 20 besitzt. Als Compiler diente der Turbo C Compiler Version 1.0. Die erhaltenen Werte sehen Sie in Tabelle 2.
Der Befehl »pokeb(0xb800,offset + + ,char)« brauchte dafür allerdings nur ganze 0,06 Sekunden. Da auch ein zusätzliches Farbattribut bei »write« und »writestr« keine Zeitveränderung erbrachte, empfiehlt sich die Benutzung dieser Ausgabefunktionen, sofern man mit »pokeb« arbeiten kann oder will. Die Befehle »write« wie auch »puts« müssen Sie jedoch jedesmal positionieren.
Man kann auf »printf« und so weiter allerdings nicht verzichten, da diese Funktionen ganz ausgezeichnete Formatierungsmöglichkeiten anbieten, über die weder »put« noch »write« verfügen. Sie sollten jedoch die Benutzung auf das minimal Erforderliche beschränken, wenn Sie schnelle Ausgaben wünschen. Die Befehle »write« und »writestr« dagegen bieten Ihnen wiederum ein zusätzliches Farbattribut an, beispielsweise »writestr ("string",244)«. Sie können so auf einfache Art Farbeffekte erzielen. Mancher wird vielleicht einwenden, daß es doch bei der Ausgabe von nur wenigen Zeilen in einem Programm völlig unerheblich ist, ob diese Ausgabe nun 0,3 Sekunden oder 0,5 Sekunden dauert. Das ist richtig, aber bei Programmen, die die Bildschirmausgabe intensiv nutzen, summieren sich diese kleinen Unterschiede zu relevanten Werten. Und wer ist nicht schon einmal ungeduldig geworden, wenn sich der Bildschirm langsamer aufbaut, als erwünscht. (Jürgen Hochstädt/gh)

## Cursorgesteuerte Programmauswahl

Haben Sie auch schon mit dem Gedanken gespielt, sich ein einfaches cursorgesteuertes Menü zu programmieren, mit dem Sie die wichtigsten Programme direkt aufrufen, ohne mit GEM, Windows oder mit MS-DOS herumhantieren zu müssen? Ein Beitrag in der Ausgabe 3'88 der DOS International, Seite 80 diente dabei als Grundlage. Die Header-Datei »cursor.h« (Listing 3) wurde für den EGA-Monitor umgeändert, während »pulldown.c« als Basis diente.

**Compiler Modell »small«**

| optimiert | Laufzeit, Messstrecke | Programm.exe, Byte |
|---|---|---|
| Size | 2,71 Sek. | 17230 |
| Size, Register | 2,71 Sek. | 17166 |
| Size, Register, Jump | 2,71k Sek. | 17070 |
| Speed | 1,82 Sek. | 17470 |
| Speed, Register | 1,82 Sek. | 17422 |
| Speed, Register, Jump | 1,82 Sek. | 17326 |
| **Compiler Modell »compact«** | | |
| Size | 2,70 Sek. | 18976 |
| Size, Register | 2,71 Sek. | 18912 |
| Size, Register, Jump | 2,71 Sek. | 18832 |
| Speed | 1,83 Sek. | 19152 |
| Speed, Register | 1,82 Sek. | 19088 |
| Speed, Register, Jump | 1,83 Sek. | 18992 |

**Tabelle 1. Laufzeitverhalten und Programmgröße**

| Funktion | Laufzeit | Header |
|---|---|---|
| printf("%c",char) | 5,01 Sek. | stdio.h |
| printf("%s",strg) | 4,74 Sek. | stdio.h |
| write("A") | 2,21 Sek. | cursor.h (siehe DOS International 3/88) |
| writestr(strg) | 4,24 Sek. | cursor.h |
| putch(char) | 3,64 Sek. | conio.h (stdio.h) |
| puts(strg) | 5,45 Sek. | stdio.h |

**Tabelle 2. Laufzeiten der verschiedenen Header-Dateien**

Falls Sie die Datei »cursor.h« schon besitzen, müssen Sie nur die Funktionen »biosequip()« und »biosmemory()« entfernen. Sie können die Farbwerte von 112 auf 0 und von 7 auf 106 ändern. Die im Listing »menuwahl.c« (Listing 4) angegebenen Farbwerte 106 sind an sich unnötig. Sie dienen lediglich zum besseren Verständnis. In »pulldown.c« setzen Sie den Zeilenabstand auf 2 (»tab..j*320«), so daß Sie über eine freie Zeile zwischen den einzelnen Menüpunktreihen haben, die vom Cursor übersprungen werden. Der Befehl »pokeb« adressiert jetzt 0xb800 statt 0xb000 gemäß der Grafikkarte. Danach wird der Cursorblock eingerahmt und das Datum und die Uhrzeit ausgegeben. Damit man aus der geschachtelten Schleife in »pulldown()« herausspringen kann, ist der Befehl »goto« eingebaut, damit man die Zeit jede Sekunde neu ausgeben kann. Dadurch ist es auch möglich, daß man den Status »warten auf Cursortaste« nicht aufgibt. Das Programm übergibt mit »exit(i)« die entsprechende Zahl (hier 1 bis 12) an die Batchdatei, in der beispielsweise mit »if errorlevel 12« die Programmaufrufe an MS-DOS gegeben werden. Die Batchdatei (Listing 5) rufen Sie mit Hilfe der

```
[3P5]    0: union REGS Register;
[000]    1:
[244]    2: void clrscr(x)
[VP1]    3: int x;
[V30]    4: {
[NE3]    5:   if (x!=112) x=7;
[UY4]    6:   Register.h.ah = 6;
[8Z4]    7:   Register.h.al = 0;
[015]    8:   Register.h.bh = x;
[C25]    9:   Register.x.cx = 0;
[545]   10:   Register.h.dh = 24;
[855]   11:   Register.h.dl = 79;
[VD8]   12:   int86(0x10,&Register,&Register);
[X30]   13: }
[X07]   14: /*****************************************
            ******/
[VK6]   15: void clsxy(x1,y1,x2,y2,x)
[A44]   16: int x1,y1,x2,y2,x;
[V30]   17: {
[NE3]   18:   if (x!=112) x=7;
[UY4]   19:   Register.h.ah = 6;
[8Z4]   20:   Register.h.al = 0;
[015]   21:   Register.h.bh = x;
[165]   22:   Register.h.ch = x1;
[P65]   23:   Register.h.cl = y1;
[765]   24:   Register.h.dh = x2;
[W65]   25:   Register.h.dl = y2;
[VD8]   26:   int86(0x10,&Register,&Register);
[X30]   27: }
[057]   28: /*****************************************
            *******/
[B25]   29: void gotoxy(x,y)
[GE2]   30: int x,y;
[V30]   31: {
[PY4]   32:   Register.h.ah = 2;
[SY4]   33:   Register.h.bh = 0;
[815]   34:   Register.h.dh = x;
[X15]   35:   Register.h.dl = y;
[VD8]   36:   int86(0x10,&Register,&Register);
[X30]   37: }
[4H7]   38: /*****************************************
            *********/
[E18]   39: void write(zeichen,farbe)
[YU2]   40: int farbe;
[YW3]   41: char zeichen;
[V30]   42: {
[7Z5]   43:  if (farbe!=112) farbe=7;
[9V4]   44:  Register.h.ah =9;
[NN7]   45:  Register.h.al = zeichen;
[7N6]   46:  Register.h.bl = farbe;
[D25]   47:  Register.x.cx = 1;
[SY4]   48:  Register.h.bh = 0;
[VD8]   49:  int86(0x10,&Register,&Register);
[X30]   50: }
[G28]   51: /*****************************************
            ***********/
[N88]   52: void writestr(string,farbe)
[YU2]   53: int farbe;
[GH3]   54: char *string;
[V30]   55: {
[2R4]   56:      while(*string){
[AD4]   57:      write('',farbe);
[S35]   58:      Register.h.ah = 14;
[SY4]   59:      Register.h.bh = 0;
[9W7]   60:      Register.h.al = *string++;
[VD8]   61:      int86(0x10,&Register,&Register);
[X30]   62:      }
[X30]   63:  }
```

```
[G28]   64: /****************************************
           **********/
[X58]   65: void xchar(zeichen,anzahl)
[173]   66: int anzahl;
[X87]   67: unsigned char zeichen;
[V30]   68: {
[3Y4]   69:   Register.h.ah=10;
[5W4]   70:   Register.h.bh=0;
[VM6]   71:   Register.x.cx=anzahl;
[Y87]   72:   Register.h.al=zeichen;
[VD8]   73:   int86(0x10,&Register,&Register);
[X30]   74: }
[G28]   75: /****************************************
           **********/
[S83]   76: int wherex()
[V30]   77: {
[PW4]   78:   Register.h.ah=3;
[5W4]   79:   Register.h.bh=0;
[VD8]   80:   int86(0x10,&Register,&Register);
[B07]   81:   return(Register.h.dh);
[X30]   82: }
[G28]   83: /****************************************
           **********/
[U83]   84: int wherey()
[V30]   85: {
[PW4]   86:   Register.h.ah=3;
[5W4]   87:   Register.h.bh=0;
[VD8]   88:   int86(0x10,&Register,&Register);
[V07]   89:   return(Register.h.dl);
[X30]   90: }
[D18]   91: /****************************************
           ***********/
[DP3]   92: int getcurs()
```

```
[V30]   93: {
[GE2]   94:   int x,y;
[3P5]   95:   union REGS Register;
[MY4]   96:   Register.h.ah = 0;
[7E8]   97:   int86(0x16,&Register,&Register);
[D55]   98:   x=Register.h.al;
[K45]   99:   y=Register.h.ah;
[W62]  100:   if(!x){
[2S2]  101:      switch(y){
[845]  102:       case 72: return(1);
[145]  103:       case 80: return(2);
[Z45]  104:       case 75: return(3);
[E55]  105:       case 77: return(4);
[2S6]  106:       default: return(0);}
[X30]  107:       }
[HR2]  108:       return(x);
[X30]  109: }
[D18]  110: /****************************************
           ************/
[LS4]  111: int biosequip()
[V30]  112: {
[XD8]  113:   int86(0x11,&Register,&Register);
[K27]  114:   return(Register.x.ax);
[X30]  115: }
[W28]  116: /****************************************
           ************/
[C75]  117: int biosmemory()
[V30]  118: {
[ZD8]  119:   int86(0x12,&Register,&Register);
[K27]  120:   return(Register.x.ax);
[X30]  121: }
```

**Listing 3. Die Headerdatei »cursor.h« übernimmt die Cursorsteuerung**

»autoexec.bat« auf und diese ruft wiederum »menuwahl.com« auf, so daß »menuwahl.com« bei Beendigung eines Anwenderprogrammes sofort wieder auf dem Bildschirm erscheint.

Wenn Sie Ihren Turbo-C-Compiler auf »tiny model« und »speed«-Optimierung einstellen, baut sich der Bildschirm bei Aufruf von »menuwahl.com« sehr schnell auf. Die im »tiny model« compilierten C-Programme können Sie ja bekannterweise mit »exebin.com« von Exe- in Com-Dateien umwandeln, und damit verringert sich auch die Laufzeit etwas. Bei der Programmierung wurde bewußt auf einige Turbo-C-Funktionen verzichtet und die Register direkt angesprochen, um das Programm portabel zu gestalten. Sie können es auch sehr leicht eigenen Wünschen anpassen, indem Sie die Menüpunkte in »menüwahl.com« und der Batchdatei entsprechend ändern. Auch hier arbeiten Sie mit der Einstellung »tiny« und erzeugen mit »exe2bin« wieder eine Com-Datei. (Jürgen Hochstädt/gh)

```
[A2R]    0: /* Programm Menuwahl. Ermoeglicht in Verbin
            dung mit einer Batchdatei eine */
[G0P]    1: /* Cursor-gesteuerte Programmauswahl. Versi
            on 1.6. Copyright (c) by DOS    */
[XEG]    2: /* International 1988. Letzte Veraenderung
            25.6.88                         */
[WSN]    3: /* Turbo-C Compiler Ver. 1.0 - Tiny Model (
            kann als Menuwahl.Com laufen)  */
[000]    4:
[TV5]    5: #include <stdio.h>
[C85]    6: #include <dos.h>
[356]    7: #include <cursor.h>
[000]    8:
[5W5]    9: /*************************************/
[1WC]   10: /* Systemzeit mit Int.21/Funktion 44 holen
            */
[000]   11:
[5N6]   12: get_zeit(std,min,sek)
[WV4]   13: int *std,*min,*sek;
[V30]   14: {
[V65]   15: union REGS inregs;
[KM5]   16: union REGS outregs;
[000]   17:
[634]   18:   inregs.h.ah = 44;
[LS7]   19:   int86(0x21,&inregs,&outregs);
[D55]   20:   *std=outregs.h.ch;
[C55]   21:   *min=outregs.h.cl;
[W45]   22:   *sek=outregs.h.dh;
[X30]   23: }
[5W5]   24: /*************************************/
[N4C]   25: /* Zeit und Datum holen und anzeigen */
[000]   26:
[5B5]   27: disp_zeit_date()
[V30]   28: {
[RC5]   29: int std,min,sek;
[V65]   30: union REGS inregs;
[KM5]   31: union REGS outregs;
[000]   32:
[MN3]   33:   gotoxy(20,21);
[AV6]   34:   writestr(" Datum =",107);
[000]   35:
[FZ9]   36:   inregs.h.ah = 42;                /* Funk
            tion 42 */
[6WC]   37:   int86(0x21,&inregs,&outregs);    /* Inte
            rrupt 21 */
[7AE]   38:   printf(" %02d.%02d.%02d   ",     /* gibt
            Datum zurueck */
[144]   39:   outregs.h.dl,
[H34]   40:   outregs.h.dh,
[HG4]   41:   outregs.x.cx);
```

```
[000]   42:
[3L6]   43:   writestr("Zeit =",107);
[HB7]   44:   get_zeit(&std,&min,&sek);
[79A]   45:   printf(" %02d:%02d:%02d ",std,min,sek);
[EC3]   46:   gotoxy(25,1);
[X30]   47: }
[5W5]   48: /*************************************/
[PE8]   49: /* Cursor spezifizieren */
[000]   50:
[C57]   51: inver(akt,breite,wert)
[M16]   52: int akt,breite,wert;
[V30]   53: {
[GM1]   54:   int i;
[16D]   55:   for(i=0;i<=breite;i+=2) pokeb(0xb800,akt+
            i,wert);
[X30]   56: }
[5W5]   57: /*************************************/
[F5E]   58: /* Cursorbewegungen festlegen und ausgeben
            */
[CKB]   59: /* jede Sekunde Zeit/Datum ausgeben */
[000]   60:
[Z5A]   61: pulldown(ze,sp,br,le,anzb,anzt)
[LH7]   62: int ze,sp,br,le,anzb,anzt;
[V30]   63: {
[6Z9]   64:  int j,i,l,max,alt_sek,std,min,sec;
[D52]   65:  char c;
[SZ4]   66:  unsigned tab[61];
[YS4]   67:  max=anzb*anzt;
[MC3]   68:  alt_sek = 0;
[WL5]   69:  ze*=160;sp*=2;br*=2;le*=2;
[EH5]   70:  for(i=1,j=0;i<=max;j++)
[2WE]   71:    for(l=0;l<anzb;l++,i++) tab[i]=ze+(j*320
            )+sp+(le*l)+(br*l)+l;
[NZ5]   72:    inver(tab[i=1],br,172);
[TG2]   73: anfang:
[T97]   74:    get_zeit(&std,&min,&sec);
[VW5]   75:    if(sec != alt_sek)  {
[0D5]   76:     disp_zeit_date();
[AF5]   77:     alt_sek = sec;      }
[000]   78:
[ZT8]   79:    if (c=kbhit() == 0) goto anfang;
[9Z6]   80:      while((c=getcurs())!=13)   {
[EN5]   81:       inver(tab[i],br,79);
[7B4]   82:       switch(c)                 {
[3AD]   83:        case 1:i=((i-anzb)<1)?i+(anzb*(anzt-
            1)):i-anzb;break;
[55E]   84:        case 2:i=(i+anzb>max)?i-(anzb*(anzt-
            1)):i+anzb;break;
[6P9]   85:        case 4:i=(i%anzb)?++i:i-anzb+1;break
            ;
```

```
[3BA]   86:         case 3:i=(i-1%anzb)?--i:i+anzb-1;bre
          ak;
[D17]   87:         default:  putchar(79); }
[YW5]   88:       inver(tab[i],br,172);
[3D3]   89:     goto anfang;
[X30]   90:                                  }
[BZE]   91:   exit(i);                          /* gibt
          Menupunkt an Batchdatei */
[X30]   92: }
[5W5]   93: /*************************************/
[5SA]   94: /* Vordere vertikale Linie ziehen */
[A86]   95: /* Zeichen 186, Farbe 106 */
[923]   96: draw_vv()
[V30]   97: {
[2G2]   98: char ld;
[4N1]   99: int m;
[755]  100:    for(m = 5;m < 19;m++) {
[853]  101:       gotoxy(m,9);
[W64]  102:       write(186,106);    }
[BC3]  103:       gotoxy(25,0);
[X30]  104: }
[5W5]  105: /*************************************/
[5KA]  106: /* Hintere vertikale Line ziehen */
[7Z2]  107: draw_vh()
[V30]  108: {
[6H2]  109: char rd;
[4N1]  110: int m;
[755]  111:    for(m = 5;m < 19;m++) {
[SC3]  112:       gotoxy(m,70);
[W64]  113:       write(186,106);    }
[BC3]  114:       gotoxy(25,0);
[X30]  115: }
[5W5]  116: /*************************************/
[PPD]  117: /* Horizontale Linien ziehen, Ecken anpasse
          n */
[073]  118: draw_horz()
[V30]  119: {
[7P1]  120: int t;
[R92]  121: char z;
[000]  122:
[853]  123:       gotoxy(4,9);
[AH3]  124:       write(201,106);
[ZT4]  125:       for(t = 10;t < 71;t++) {
[Y83]  126:       gotoxy(4,t);
[ED4]  127:       write(205,106);         }
[9H3]  128:       write(187,106);
[AD3]  129:       gotoxy(19,9);
[DH3]  130:       write(204,106);
[ZT4]  131:       for(t = 10;t < 71;t++) {
[VK3]  132:       gotoxy(19,t);
[ED4]  133:       write(205,106);         }
[7H3]  134:       write(185,106);
[1D3]  135:       gotoxy(20,9);
[8H3]  136:       write(186,106);
[ZN3]  137:       gotoxy(20,70);
[8H3]  138:       write(186,106);
[2D3]  139:       gotoxy(21,9);
[9H3]  140:       write(200,106);
[ZT4]  141:       for(t = 10;t < 71;t++) {
[LK3]  142:       gotoxy(21,t);
[ED4]  143:       write(205,106);         }
[AH3]  144:       write(188,106);
[X30]  145: }
[5W5]  146: /*************************************/
[0F1]  147: main()
[V30]  148: {
[GM1]  149: int i;
[000]  150:
[L03]  151:   clrscr(106);
[0P4]  152:   clsxy(5,11,18,68,121);
[HX4]  153:   clsxy(20,11,20,68,121);
[P43]  154:   gotoxy(3,0);
[Z53]  155:   draw_vv();
[X23]  156:   draw_vh();
[CE3]  157:   draw_horz();
[VBG]  158:   gotoxy(6,25); writestr(" IBM Turbo-XT Pro
          grammauswahl ",31);
[KU9]  159:   gotoxy(10,20); writestr("   A:\\     ",79)
          ;
[LV9]  160:   gotoxy(10,35); writestr("   C:\\     ",79)
          ;
[AV9]  161:   gotoxy(10,50); writestr("   D:\\     ",79)
          ;
[HBA]  162:   gotoxy(12,20); writestr("  X-Tree   ",79);
[GSA]  163:   gotoxy(12,35); writestr(" T-Pascal ",79);
[U8A]  164:   gotoxy(12,50); writestr("   Gem     ",79);
[E5A]  165:   gotoxy(14,20); writestr(" IBM Menu ",79);
[UZA]  166:   gotoxy(14,35); writestr(" Fontasy  ",79);
[20B]  167:   gotoxy(14,50); writestr(" Turbo-C  ",79);
[4GA]  168:   gotoxy(16,20); writestr(" Ear-Mike ",79);
[3ZA]  169:   gotoxy(16,35); writestr(" CopyIIIp ",79);
[DMA]  170:   gotoxy(16,50); writestr("  Spega    ",79);
[HN3]  171:   gotoxy(24,10);
[ULM]  172:   writestr(" Auswahl: Cursortasten - Progra
          mm: <RETURN> - Ende: Ctrl_C ",71);
[HC3]  173:   gotoxy(25,2);
[L63]  174:   get_zeit();
[0D5]  175:   disp_zeit_date();
[7T5]  176:   i=pulldown(10,20,9,6,3,4);
[X30]  177: }
```

**Listing 4. Das Hauptprogramm »menuwahl.com«**

## Systeminformationen mit Turbo C

Auch in C können Sie Systeminformationen ohne großen Aufwand erhalten. Speziell Turbo C bietet Ihnen ausgezeichnete Funktionen für die systemnahe Programmie-

```
[3H2]    0: echo off
[7X2]    1: :schleife
[6Y2]    2: menuwahl
[MM0]    3: cls
[VE8]    4: if errorlevel 12 goto punkt12
[PE8]    5: if errorlevel 11 goto punkt11
[HE8]    6: if errorlevel 10 goto punkt10
[X38]    7: if errorlevel 9 goto punkt9
[L38]    8: if errorlevel 8 goto punkt8
[938]    9: if errorlevel 7 goto punkt7
[Z28]   10: if errorlevel 6 goto punkt6
[N28]   11: if errorlevel 5 goto punkt5
[B28]   12: if errorlevel 4 goto punkt4
[128]   13: if errorlevel 3 goto punkt3
[R18]   14: if errorlevel 2 goto punkt2
[D18]   15: if errorlevel 1 goto punkt1
[URK]   16: echo *** Errorlevel wird vom Programm falsc
          h zuruekgegeben ***
[PK1]   17: pause
[634]   18: goto schleife
[RK2]   19: :punkt1
[9E0]   20: A:\
[9T0]   21: cd ..
[XK2]   22: goto end
[634]   23: goto schleife
[YK2]   24: :punkt2
[BE0]   25: C:\
[ZH0]   26: cd\
[XK2]   27: goto end
[634]   28: goto schleife
[5L2]   29: :punkt3
[CE0]   30: D:\
[9T0]   31: cd ..
[634]   32: goto schleife
[CL2]   33: :punkt4
[YU2]   34: C:\xtree
[634]   35: goto schleife
[LL2]   36: :punkt5
[CE0]   37: D:\
[F13]   38: cd tpascal
[7L1]   39: turbo
[ZN0]   40: cd\
[SF3]   41: D:\pulldown
[634]   42: goto schleife
[UL2]   43: :punkt6
[CE0]   44: D:\
[SL0]   45: gem
[ZH0]   46: cd\
[634]   47: goto schleife
[1M2]   48: :punkt7
[BE0]   49: C:\
[N21]   50: menu
[ZN0]   51: cd\
[SF3]   52: D:\pulldown
[634]   53: goto schleife
[8M2]   54: :punkt8
[CE0]   55: D:\
[0C1]   56: cd fy
[SA0]   57: fy
[ZH0]   58: cd\
[634]   59: goto schleife
[FM2]   60: :punkt9
[CE0]   61: D:\
[1V2]   62: cd turboc
[U90]   63: tc
[ZH0]   64: cd\
[634]   65: goto schleife
[RK2]   66: :punkt10
[BE0]   67: C:\
[GL2]   68: cd dbase
[GG1]   69: dbase
[ZH0]   70: cd\
[634]   71: goto schleife
[RK2]   72: :punkt11
[CE0]   73: D:\
[HT2]   74: cd copy
[RY2]   75: copyiiip
[ZH0]   76: cd\
[634]   77: goto schleife
[RK2]   78: :punkt12
[CE0]   79: D:\
[3G1]   80: spega
[634]   81: goto schleife
[RE1]   82: :ende
[634]   83: goto schleife
[YY0]   84: :end
[MM0]   85: cls
```

**Listing 5. Diese Batchdatei müssen Sie in Ihre »autoexec.bat« einbauen**

rung. Einige davon können Sie in »checksys.c« sehen. Eigentlich mehr als unbedingt nötig, um zu demonstrieren, daß häufig auch verschiedene Wege zum Ziel führen können.

Das Programm (Listing 6) ist bewußt ausführlich kommentiert, um zusätzliche Erklärungen zu ersparen. Selbstverständlich läßt sich auch dieses Programm nicht in allen Bereichen universell einsetzen, da die Ablegung der Informationen wie Bios-Copyright, Gerätehersteller und so weiter leider nicht genormt sind. Die erste Zeile kann bei Ihrem Computer anders ausfallen. Erfahrungsgemäß sind aber diese Angaben im Speicherbereich FE00:0000hex bis FE00:0050hex abgelegt. Leider teilweise mit »CR/LF« zwischen den einzelnen Daten, so daß Sie die Werte »k« und »gotoxy()« eventuell entsprechend anpassen müssen.

Ob eine Echtzeituhr, auch »RTC« oder »Real Time Clock« genannt, vorhanden ist, erfährt man durch Bit 3, Funktion 44, Interrupt 21, wenn Sie das Basisregister BX auf 800hex setzen. Das Ausmaskieren eines Bits mit den Shiftoperatoren »< <« (shift links) und »> >« (shift rechts) ist sehr einfach. Man muß nur darauf achten, daß mit Bit 0 das äußerste rechte Bit bezeichnet ist und mit Bit 7 – bei halbem Registerpaar (also AL, AH und so weiter – beziehungsweise Bit 15 – bei vollem Registerpaar (AX, BX) – das äußerste linke Bit. »get_byte()« speichert während der Abarbeitung von »getfat()« das aktuelle Laufwerk, da alle Laufwerke ab Laufwerk C: \ angewählt und damit für MS-DOS jeweils aktuell sind. Die Rückgabe erfolgt durch den Befehl »setdisk()«.

Die Headerdatei »cursor.h« (siehe DOS, Ausgabe 3'88, Seite 80) müssen Sie auch hier etwas ändern, falls Sie mit einer anderen Grafikkarte arbeiten. Benutzer von Monochrom-Monitoren sollten die Originaldatei einbinden, oder alle Writestr-Befehle durch »printf« ersetzen und die entsprechenden Farbangaben im Listing entfernen. Die Farben sind auf roten Hintergrund, grünen (106) beziehungsweise hellblauen Text (107) und weißen Rahmen (121) eingestellt. Diese Farbabsetzungen ersparen bei der Programmierung häufig das umständliche Ziehen von Linien und Boxen für die Bildschirmaufteilung. »bios-equip()« arbeitet mit dem Interrupt 11 und ist ebenfalls ausführlich in DOS 3'88 erläutert.

In der Annahme, daß das erste Festplattenlaufwerk wohl überwiegend mit C bezeichnet ist, ist im Programm die doch etwas aufwendige Einzelabfrage und Anwahl der Diskettenlaufwerke A und B unterlassen. Man hätte sonst auch noch mit Fehlerroutinen arbeiten müssen, da man bei der Anwahl den Interrupt 24 auslösen kann und Sie damit die Bildschirmausgabe durch MS-DOS-Fehlermeldungen unterbrechen. Ebenso hat sich die Berechnung des Speichers unterschiedlich gestaltet. Bei »getfat()« erzwingt der Cast-Operator eine Operation mit Float-Werten. Da die Struktur von »getfat()« bereits in »fatinfo« festgelegt ist, müssen Sie hier zum Cast-Operator greifen. Bei »getdfree()« ist die Struktur »dfree« zwar auch schon festgelegt, Sie können es aber dabei belassen und kommen auch nicht über die zulässigen Integerwerte hinaus, wenn Sie eine bestimmte Reihenfolge in der Berechnung einhalten. Es mag etwas verwirren, daß einerseits durch 1024, andererseits durch 1000 dividiert wird, aber die Ausgabe in KBytes ist in beiden Fällen korrekt. Um die Bildschirmausgabe nicht zu überladen, wurde auf weitere Informationsausgaben verzichtet und auch keine Kommandozeile mit Parameterübernahme benutzt. Sie können aber das Programm leicht erweitern. Vergessen Sie nicht, es mit der Compilereinstellung »tiny« zu compilieren und daraufhin mit »exe2bin« in eine Com-Datei umzuwandeln. Die Warnungen können Sie ignorieren.

(Jürgen Hochstädt/gh)

```
[VGG]    0:     /* CHECKSYS.C  - Copyright (c) J.Hochst
           aedt  Juni 1988 - Vers. 1.4 */
[8AN]    1:     /* Headerdatei cursor.h aus DOS Interna
           tional 3/88, etwas veraendert */
[FWH]    2:     /* Compiled mit Turbo C Compiler  Versi
           on 1.0  - Tiny model - */
[000]    3:
[095]    4: #include <bios.h>
[GLL]    5: #include <dos.h>                          /* cur
           sor.h nicht unbedingt notwendig */
[S0M]    6: #include <dir.h>                          /* abe
           r empfehlenswert bei Farbmonitor*/
[SHL]    7: #include <cursor.h>                       /* da
           Farbeffekte leicht zu erzielen */
[000]    8:
[000]    9:
[F33]   10: get_byte()
[V30]   11: {
[RAF]   12:     union REGS inregs;                    /* Auf
           ruf Int. 21 - Funktion 44 */
[5RH]   13:     union REGS outregs;                   /* Abf
           rage Echtzeituhr vorhanden */
[000]   14:
[WA4]   15:     inregs.h.ah = 0x44;
[P24]   16:     inregs.h.al = 00;
[SU4]   17:     inregs.x.bx = 0x0800;
[LS7]   18:     int86(0x21,&inregs,&outregs);
[X30]   19: }
[000]   20:
[000]   21:
[NLF]   22: get_nbyte()                              /* Akt
           uelles Laufwerk merken */
[V30]   23: {
[V65]   24:     union REGS inregs;
[KM5]   25:     union REGS outregs;
[V93]   26:     unsigned a;
[4P3]   27:     a = getdisk();
[000]   28:
[3B4]   29:     inregs.h.ah = 0x19;
[KB4]   30:     inregs.h.al = a;
[LS7]   31:     int86(0x21,&inregs,&outregs);
[X30]   32: }
[000]   33:
[MVK]   34: struct dfree frei;                       /* lie
           st freien Speicher auf Laufwerk */
[P4K]   35: unsigned char drive;                     /* ben
           utzt bei: Aktuelles Lw.: ...     */
[000]   36:
[46K]   37: struct fatinfo free;                     /* lie
           st FAT aus und gibt totalen      */
[4CH]   38: unsigned char drv;                       /* Spe
           icher + ID_Byte zurueck          */
[000]   39:
[0F1]   40: main()
[V30]   41: {
[SW8]   42:  int k,z,sektor,cluster,total;
[D2C]   43:  unsigned uhr,i,tot,fr,clus,sec,q,m,kbv;
[L82]   44:  char t;
[NZ7]   45:  char a[] ="EGA/CGA Farbe 80 x 25";
[LZ7]   46:  char b[] ="EGA/CGA Farbe 40 x 25";
[XG8]   47:  char c[] ="Monochrome    80 x 25";
[VX8]   48:  char g[] ="Graphikkarte    : ";
[5PB]   49:  char ev[] ="Echtzeituhr     :  vorhanden";
[H5A]   50:  char ek[] ="Echtzeituhr     :  keine";
[000]   51:
[YYA]   52:  q = get_nbyte();                        /* Akt
           . LW merken */
[E6F]   53:  m = setdisk(getdisk());                 /* Wie
           viel moegliche LW ? */
[0RC]   54:  z = getdisk();                          /* Wel
           ches ist akt. LW ? */
[5FE]   55:  i = biosequip();                       /* s. DO
           S International 3/88 */
[000]   56:
[YXF]   57:   clrscr(121);                           /* in
           cursor.h - EGA Farben von 00 bis */
[LGF]   58:   clsxy(0,5,24,73,106);                  /* 255
            waehlbar, ab 128 blinkend       */
[SC3]   59:   gotoxy(0,14);
[000]   60:
[T9R]   61:   /* Geraetinformation auslesen. Ohne Unter
           teilung sind es 2 oder 3 Zeilen */
[LCP]   62:   /* Das Druckbild und die ausgelesenen Inf
           ormationen sind  Hersteller */
[1EP]   63:   /* spezifisch und muessen eventuell entsp
           rechend angepasst werden */
[HLE]   64:   for(k = 0x0000; k < 0x0015; k++)  printf(
           "%c", peekb(0xfe00,k));
[2D3]   65:   gotoxy(0,36);
[0NE]   66:   for(k = 0x0016; k < 0x0037; k++)  printf(
           "%c", peekb(0xfe00,k));
[453]   67:   gotoxy(3,7);
[000]   68:
[YM6]   69:   /* in dos.h enthalten */
[33M]   70:   printf(" Betriebssystem  :  DOS Vers. %d.
           %d  \n",_osmajor,_osminor);
[000]   71:
[9L9]   72:    /* Ist Echtzeituhr vorhanden ? */
[EG4]   73:   uhr = get_byte();
[WVD]   74:   if(uhr << 4 >> 7 == 0) { gotoxy(10,8); wr
           itestr(ev,107); }
```

```
[T2B]   75:                  else    { gotoxy(10,8); wr
          itestr(ek,107); }
[000]   76:
[TNL]   77:   /* biosequip() ist in cursor.h und in bio
          s.h enthalten. Mit << >> wird */
[9WB]   78:   /* das entsprechende Bit ausmaskiert */
[N88]   79:   if ((i << 10 >> 14) == 2) { gotoxy(4,7);
[44B]   80:                               printf(" %s %
          s   \n",g,a); } /* Videomodus */
[N88]   81:   if ((i << 10 >> 14) == 3) { gotoxy(4,7);
[1XA]   82:                               printf(" %s %
          s   \n",g,c); } /* auslesen   */
[N88]   83:   if ((i << 10 >> 14) == 1) { gotoxy(4,7);
[CN6]   84:                               printf(" %s %
          s   \n",g,b); }
[ANH]   85:   gotoxy(5,7,121); printf(" Videomodus
            :  %d \n",peekb(0x40,0x49));
[P58]   86:   if ((i << 3 >> 15) == 1)  { gotoxy(8,7);
[NSC]   87:                               writestr(" Ga
          me Port       :  vorhanden ",107);}
[F58]   88:   if ((i << 3 >> 15) == 0)  { gotoxy(8,7);
[P9B]   89:                               writestr(" Ga
          me Port       :  keiner ",107);}
[V98]   90:   if ((i << 14 >> 15) == 0) { gotoxy(9,7);
[5BC]   91:                               writestr(" Co
          prozessor     :  keiner ",107);}
[V98]   92:   if ((i << 14 >> 15) == 1) { gotoxy(9,7);
[4UD]   93:                               writestr(" Co
          prozessor     :  vorhanden ",107);}
[000]   94:
[HSN]   95:   /* Disklaufwerk vorhanden ?  Wenn ja, wie
          viele. Vorsicht, bei manchen */
[7NM]   96:   /* Geraeten werden alle Laufwerke (nicht
          nor Floppys) ausgegeben */
[BR4]   97:   if ((i << 15 >> 15) == 1) {
[B88]   98:   if ((i << 8 >> 14) == 0) { gotoxy(14,7);
[ZFB]   99:                               writestr(" Dis
          kettenlaufw. :  1",106);}
[K88]  100:   if ((i << 8 >> 14) == 1) { gotoxy(14,7);
[6GB]  101:                               writestr(" Dis
          kettenlaufw. :  2",106);}
[T82]  102:                             } else {
[H2M]  103:          gotoxy(14,7); writestr(" Diskettenl
          aufw. :  nicht vorhanden ",106);}
[000]  104:
[B0K]  105:   /* Wieviele von jeder Standard-Schnittste
          lle vorhanden */
[MUD]  106:    gotoxy(12,7); printf(" Parallel Port(s):
           %d ", i >> 14);
[7DF]  107:    gotoxy(12,31); printf(" Serielle Port(s):
           %d ", i << 4 >> 13);
[000]  108:
[BHL]  109:   /* Festplatten auslesen. Erste HD C:\. Co
          de aendern, falls anders */
[2K8]  110:   for(drv = 3;drv < m;drv++) {
[035]  111:     getfat(drv,&free);
[YCB]  112:          if(free.fi_fatid << 13 >> 15 != 0)
          continue;
[CX1]  113:          else {
[MF7]  114:   sektor = free.fi_bysec;
[1A8]  115:   cluster = free.fi_sclus;
[WP6]  116:   total = free.fi_nclus;
```

```
[MTL]  117:   kbv = ((((unsigned)sektor*(unsigned)clust
          er)/1024)*(unsigned)total);
[YX3]  118:   switch(drv)  {
[HDG]  119:   case 3: gotoxy(15,7); printf(" Festplatte
          (n)   :  C:\\ %5d kB ",kbv);
[KV1]  120:            break;
[PEB]  121:   case 4: gotoxy(15,40); printf(" D:\\ %5d
          kB ",kbv);
[KV1]  122:            break;
[SFB]  123:   case 5: gotoxy(15,56); printf(" E:\\ %5d
          kB ",kbv);
[KV1]  124:            break;
[4VL]  125:   default: gotoxy(23,7); printf(" Quelldate
          i um n LW erweitern \n");
[3B3]  126:                 } /* switch */
[3E2]  127:               } /* else */
[665]  128:             } /* for Schleife */
[7XG]  129:    setdisk(q);                          /*
          Anfangs-LW wieder einstellen */
[000]  130:
[ECF]  131:   /* biosmemory() in cursor.h und bios.h en
          thalten */
[S2H]  132:   gotoxy(16,7); printf(" RAM Speicher    :
           %d kByte ",biosmemory());
[000]  133:
[U1R]  134:   /* Daten fuer aktuelles Laufwerk auslesen
          . Getdfree() in dos.h enthalten */
[0X6]  135:   getdfree(drive,&frei);
[4G5]  136:   tot = frei.df_total;
[PC5]  137:   fr = frei.df_avail;
[1S5]  138:   clus = frei.df_sclus;
[MX4]  139:   sec = frei.df_bsec;
[5B2]  140:   drive = 0;
[000]  141:
[3D3]  142:   gotoxy(18,7);
[6LG]  143:   printf(" Aktuelles Lw.:  Noch %u von tota
          l %u Cluster frei",
[7H6]  144:            fr,tot); gotoxy(19,7);
[G79]  145:   printf(" =  %u kByte  von  %u kByte ",
[0DD]  146:            ((sec*clus)/1000)*fr,((sec*clus)
          /1000)*tot); gotoxy(20,7);
[M1K]  147:   printf(" Ein Cluster besteht aus %u Sekto
          ren zu je %u Bytes. ",
[DM2]  148:            clus, sec);
[U1S]  149:   /* Vorsicht bei der Ausrechnung. Bei Vers
          chiebung der Einzelwerte voellig */
[E0R]  150:   /* unterschiedliche Resultate. Auch bei a
          nderer Typdekl. der Variablen */
[000]  151:
[H7M]  152:   /* Welches ist aktuelles Laufwerk. Setdis
          k() in dir.h enthalten */
[DVA]  153:   if(z == 0) t='A'; if(z == 1) t='B'; if(z
          == 2) t='C';
[14B]  154:   if(z == 3) t='D'; if(z == 4) t='E'; gotox
          y(22,7);
[8PH]  155:   printf(" Aktuelles Laufwerk : %c  -  Moeg
          l. logische Laufw.: %d ",
[EB4]  156:            t, setdisk(z));
[000]  157:
[CV2]  158: } /* main() */
```

**Listing 6. »checksys.com« gibt Ihnen Systeminformationen aus**


## *Wissen Sie mehr von PCs als andere Leute?*

Dann schreiben Sie uns. Die Redaktion DOS International sucht aktive Mitarbeiter, die über interessante Gebiete aus der »Computerei« berichten wollen. MS-DOS, DOS Plus, GEM, Drucker, Basteleien – es gibt nichts, worüber nicht andere Leser unserer Zeitschrift auch etwas lesen wollen.

Ihren Einsatz lassen wir uns natürlich etwas kosten. Ihr Manuskript wird selbstverständlich honoriert. Bilder, Zeichnungen, Tabellen und Listings werden gesondert berechnet.

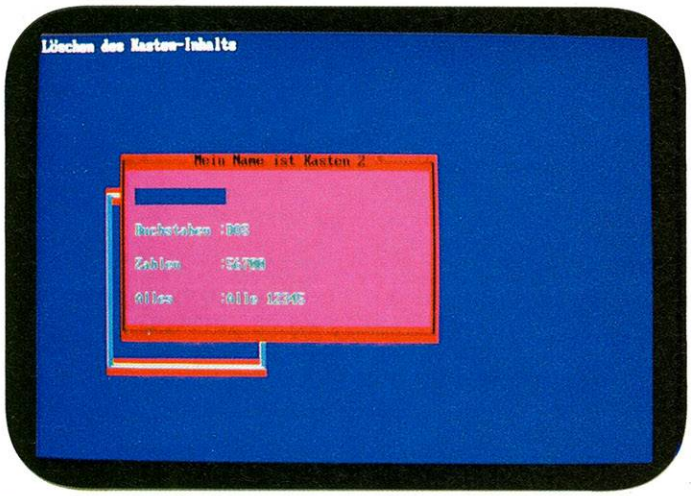
Doch bevor Sie sich hinsetzen und das Schreiben anfangen: Rufen Sie uns an. Sagen Sie uns, was Sie schreiben wollen, und wir klären ob – und wann – der Artikel in unser Konzept paßt.

Der Artikel muß mit einem Textsystem erfaßt werden. Wir brauchen von Ihnen eine Diskette im MS-DOS-Format, auf der der Text als reines ASCII-Feld steht.

Schreiben Sie uns auch, wenn Sie nur Vorschläge für Themen haben. DOS International ist nämlich eine Zeitschrift von aktiven PC-Besitzern für aktive PC-Benutzer. Besonders das DOS-Forum lebt von Ihrer Mitarbeit. Auch die Redaktion kann nicht alle Antworten wissen. Fragen an uns, die nicht sofort beantwortet werden können, werden im DOS-Forum veröffentlicht. Wissen Sie eine Antwort, so wird diese veröffentlicht.

Sie sehen: DOS International ist die PC-Zeitschrift, in der das Wissen der PC-Benutzer untereinander ausgetauscht wird.

***Redaktion · DOS International · Hans-Stießberger-Str. 3 · 8013 Haar/München · Tel.: (089) 46 10 01-0***




**Ein Fenster mit der Eingaberoutine**

# Fenstertechnik mit Turbo C

**Ohne Fenstertechnik fehlt Programmen der gewisse »Touch«. Innerhalb kurzer Zeit hat sich diese Art der Programmführung aufgrund ihrer Benutzerfreundlichkeit durchgesetzt, so daß kein komfortables Programm mehr auf diese Bildschirmgestaltung verzichten kann.**

Fenstertechnik, das bedeutet Komfort in der Bedienung. Das Programm »kasten.c« (Listing 1) stellt nun auch für alle Turbo-C-Programmierer eine Möglichkeit dar, einfach und schnell Fenster auf den Bildschirm zu zaubern. Die mit »kasten.c« programmierbaren Fenster haben das gleiche Aussehen wie die Fenster des Turbo-C-Compilers (siehe Bild). Mit Hilfe einiger weniger Änderungen erhalten jedoch die Fenster auch ein ganz anderes Aussehen. »kasten.c« erkennt automatisch, welche Grafikkarte Sie benutzen. Der Aufbau der Windows erfolgt sehr schnell und völlig ohne »Schneegestöber«. Die maximale Anzahl der gleichzeitig darstellbaren Windows ist nur vom Speicher abhängig, und die maximale Größe eines Windows reicht über eine volle Bildschirmseite.
Wenn Sie die Fenster in eigene Anwendungen einbauen wollen, müssen Sie nur die Datei »kasten.h« (Listing 2) mit der Anweisung »#include« in das eigene Programm aufnehmen und das Modul »kasten.obj« dazu binden. Die dazu notwendige Projekt-Datei ist unter dem Namen »test.prj« (Listing 3) aufgeführt. Ein Beispiel für die Möglichkeiten dieser Applikation bietet Ihnen das Programm »test.c« (Listing 4). Anwendungen, die mit »kasten.c» arbeiten, können Sie mit den Speichermodellen »small« und »tiny« bearbeiten.

## Bereits standardmäßig ist ein doppelter Rahmen definiert

Bei der Programmierung der Windows müssen Sie selbst Sorge dafür tragen, daß Sie immer nur das oberste Fenster bearbeiten. Dies gilt für alle Funktionen. Im weiteren erfahren Sie Grundsätzliches über diese Funktionen:
– Zeiger auf die Fenster: Für jedes Window müssen Sie einen Zeiger vom Typ »Kasten« definieren. Die Definition von »Kasten« befindet sich in der Datei »kasten.h«.
– Umrahmung der Windows: In der globalen Variablen »rand[]« sind die Zeichen für die Umrahmung der Fenster gespeichert. Diese Zeichen können Sie während des Programmlaufs beliebig ändern, und somit den Fenstern ein unterschiedliches Aussehen verleihen. Die Standardeinstellung ist ein Rand mit Doppelstrich. Der Aufruf der verschiedenen Funktionen geschieht folgendermaßen:

**ka__auf:** Öffnen eines Fensters
Syntax : ka__auf(x,y,l,b,ra,ia)
int x,y,l,b,ra,ia;
x,y – Anfangskoordinaten des Windows
l,b – Länge/Breite des Windows
ra – Attribut für den Rand (Hintergrund << 4| Vordergrund)
ia – Attribut für das Window - " -
Die Funktion gibt entweder einen Zeiger auf den »alten« Hintergrund oder aber »0« aus, wenn kein Speicherplatz mehr vorhanden ist. Der Cursor steht in der ersten Zeile und Spalte des Windows, ist jedoch unsichtbar.

**ka__aus:** Ausschalten eines Windows
Syntax : ka__aus(ka);
Kasten ka;
ka – Zeiger auf den Kasten.
Diese Funktion liefert keinen Wert zurück. Beim Ausschalten des letzten Windows wird der Cursor wieder aktiviert.

**ka__titel:** Einfügen eines Titels im oberen Rand
Syntax : ka__titel(ka,str,ra);
Kasten ka;
char str[];
int ra;
ka – Zeiger auf den Kasten
str – String
ra – Attribut für den Text (Hintergrund << 4| Vordergrund)
Die Funktion liefert den Wert »-1«, wenn der String zu lang ist, ansonsten den Wert 0.

**ka__leer** – löscht den Inhalt eines Fensters
Syntax : ka__leer(ka);
Kasten ka;
ka – Zeiger auf den Kasten:
Diese Routine liefert keinen Wert zurück.

**ka__lies:** Einlesen eines Strings aus einem Window
Syntax : ka__lies(ka,sptr,art,len);
Kasten ka;
char *sptr, art;
int len;
ka – Zeiger auf einen Kasten:
sptr – char – Variable
art – »a« oder »A« für Buchstaben
»n« oder »N« für Zahlen und die Zeichen » + «, » – ae- ,»/«
»x« oder »X« für »alles«
len – Länge des einzulesenden Strings
Diese Länge des Strings kann maximal die Breite des Windows annehmen. Mit der Taste [Return] oder [Esc] können Sie eine Eingabe abschließen. Bei falschen Eingaben ertönt ein Warnton. Während dieser Funktion ist der Cursor sichtbar.

**ka__schreib:** Schreiben eines Strings in ein Fenster
Syntax : ka__schreib(ka,str,attr);
Kasten ka;
fchar *str;
int attr;
ka – Zeiger auf einen Kasten
str – zu schreibender String
attr – Attribut: bei »-1« Attribut wie das Window selbst
Der String wird bis zur maximalen Länge des Fensters ausgegeben. Diese Funktion gibt keinen Wert zurück.

**ka__cpos:** Cursor positionieren
Syntax : ka__cpos(ka,cx,cy);
Kasten ka;
int cx,cy;
ka – Zeiger auf einen Kasten
cx,cy – neue Cursorposition

Es erfolgt eine Überprüfung der Werte. Die Funktion liefert »-1«, wenn die Werte ungültig sind, sonst 0.

Hier ein kurzer Einblick in die Arbeitsweise des Programms. Wenn Sie eine Bildschirmseite mit dem Befehl »printf« eingeben, erkennen Sie sofort, daß sich auf diesem Weg kein Window programmieren läßt. Es ist also notwendig, direkt in den Bildschirmspeicher zu schreiben. Um dies zu bewerkstelligen, müssen Sie zunächst feststellen, welcher Monitortyp angeschlossen ist und in welchem Modus sie sich befinden; das heißt, Sie müssen abklären, welche Speicherkonfiguration vorliegt. Diese Aufgabe erledigt in »kasten.c« die Funktion »get_basis()«. Sie fragt die entsprechenden Speicherzellen ab und setzt so die Startadresse für den Bildschirmspeicher. Darüber hinaus stellt das Programm noch fest, welchen Port (beispielsweise »0x3da« für Farbe oder »0x3ba« für Monochrom) Sie benutzen. Bei nicht hundertprozentig kompatiblen Computern funktioniert dies nicht. Das liegt daran, daß diese Computer andere (jedoch auch) konstante Werte liefern. Als (mögliche) Lösung bietet sich hier das Ausprobieren an. Wenn nicht mit verschiedenen Grafikmodi gearbeitet wird, läßt sich die Funktion »get_basis()« folgendermaßen umschreiben (Listing 5) (im Monochrom-Modus sollten Sie dann allerdings auf den Aufruf »warte()« verzichten).

Eine weitere Möglichkeit bietet Ihnen die Funktion »bios-equip()«. Damit können Sie wiederum den Modus bestimmen. Diese Routine befindet sich auch in der Datei »get_b_2.c« (Listing 6).

Das Schreiben (egal in welche Speicherzelle) erfolgt mit Hilfe von Pointern. Da die Adresse des Bildschirmspeichers entweder bei »0xb8000« für Farbe oder bei »0xb0000« für Monochrom liegt, können Sie diese Adressen nicht mit den normalen Pointern erzeugen. Es gibt jedoch in Turbo C die Funktion »mk_fp« (bedeutet »make far pointer«) zur Generierung eines »far pointers«. Mit deren Hilfe können Sie nun den entsprechenden Speicherbereich erreichen.

Als Programmierer müssen Sie nur noch darauf achten, daß Sie immer nur das oberste Window bearbeiten, da Sie sonst die Fenster zerstören könnten.

(Sascha Kerres/gh)

```
[F6B]    0: /*****************************************
            ***********************************
[CN8]    1: **                                     **
[W2K]    2: ** KASTEN.C - Funktionen zur Darstellung vo
            n Windows                             **
[CN8]    3: **                                     **
[H7F]    4: ** für den TURBO-C - Compiler von BORLAND V
            1.0                                   **
[CN8]    5: **                                     **
[L5D]    6: ** von Sacha Kerres, März 1988
                                                  **
[CN8]    7: **                                     **
[87B]    8: *****************************************
            ***********************************/
[000]    9:
[C85]   10: #include <dos.h>
[VS5]   11: #include <alloc.h>
[000]   12:
[DKA]   13: #define WARTE() while(inportb(port)&1)
[000]   14:
[S17]   15: extern int get_basis();
[3K5]   16: extern void c_pos();
[VF6]   17: extern void cur_ein();
[YG6]   18: extern void cur_aus();
[000]   19:
[ZXC]   20: unsigned char rand[8] = {200,201,205,205,18
            6,186,187,188};
[000]   21:
[FBE]   22: static int port;
                          /* BS-Adressen */
[W35]   23: static int basis;
[000]   24:
[R9G]   25: static int anzahl = 0;
                 /* Anzahl offener Kasten */
[000]   26:
[D8G]   27: /********************** KA_AUF - Öffnen ei
            nes Kastens **********************
[6L8]   28: *                                     *
[SAL]   29: * liefert einen far-Pointer auf einen Kaste
            n zurück                              *
[WRK]   30: * oder NULL wenn zu wenig Speicher vohanden
            ist                                   *
[6L8]   31: *                                     *
[87B]   32: *****************************************
            ***********************************/
[000]   33:
[349]   34: int far *ka_auf(x,y,l,b,ra,ia)
[000]   35:
[NR4]   36: int x,y,l,b,ra,ia;
[V30]   37:  {
[L7G]   38:    int far *kptr;
                  /* Pointer auf den Kasten */
[HCL]   39:    register int far *ptr;
               /* Pointer auf den Bildspeicher */
[1SD]   40:    int i,j,k;
                        /* Schleifenzähler */
[000]   41:
[NXG]   42:    unsigned zeichen;
                   /* Zeichen zum schreiben */
[000]   43:
[643]   44:    cur_aus();
[000]   45:
[K3H]   46:    ptr = MK_FP(get_basis(),0);
                  /* Pointer auf Bildschirm */
```

```
[000]   47:
[OVK]   48:    if (!basis)                /* Feststellun
            g der BS-Adressen nicht möglich */
[8Y1]   49:      exit(-1);
[BEH]   50:    kptr = farmalloc(((l+1)*(b+1)*4)+18);
                     /* Speicher reservieren */
[OY3]   51:    if (kptr == 0L)
[YTD]   52:     return(0L);
                     /* Zu wenig Speicher */
[000]   53:
[DLF]   54:    anzahl++;
             /* Anzahl der Kasten plus eins */
[000]   55:
[4LE]   56:    kptr[0] = x;
                   /* Anfangs - Koprdinaten */
[V53]   57:    kptr[1] = y;
[4ND]   58:    kptr[2] = l;
                        /* Länge / Breite */
[243]   59:    kptr[3] = b;
[9PC]   60:    kptr[4] = ra;
                              /* Attribute */
[2L3]   61:    kptr[5] = ia;
[6VD]   62:    kptr[6] = x+1;
                        /* Cursor - Position */
[PR3]   63:    kptr[7] = y+1;
[000]   64:
[5K0]   65:    x--;
[6K0]   66:    y--;
[000]   67:
[131]   68:    k = 9;
[XGH]   69:    for (i=0;i<=l;i++)                    /*
            Speichern des alten Bildinhalts*/
[FB4]   70:      for (j=0;j<=b;j++)
[V30]   71:       {
[5R8]   72:        kptr[k++] = ptr[(x+i)*80 + y+j];
[X30]   73:       }
[000]   74:
[VRE]   75:    ra <<=8;
             /* Attribut in die ersten 8 Bit */
[XM1]   76:    ia <<=8;
[000]   77:
[V1G]   78:    zeichen = ra + rand[0];
                      /* Zeichen aufbauen */
[LKD]   79:    WARTE();
                    /* Auf Rücklauf warten */
[F8F]   80:    ptr[(x+1)*80 + y] = zeichen;
                      /* Zeichen schreiben */
[000]   81:
[1F6]   82:    zeichen = ra + rand[1];
[7M1]   83:    WARTE();
[ZU6]   84:    ptr[x*80 + y] = zeichen;
[000]   85:
[Z14]   86:    for (i=1;i<b;i++)
[V30]   87:     {
[6F6]   88:      zeichen = ra + rand[2];
[7M1]   89:      WARTE();
[DS6]   90:      ptr[x*80 + y + i] = zeichen;
[000]   91:
[BF6]   92:      zeichen = ra + rand[3];
[7M1]   93:      WARTE();
[457]   94:      ptr[(x+1)*80 + y+i] = zeichen;
[X30]   95:     }
[000]   96:
[734]   97:    for (i=1;i<l;i++)
[V30]   98:     {
[GF6]   99:      zeichen = ra + rand[4];
[7M1]  100:      WARTE();
[1H6]  101:      ptr[(x+i)*80 + y] = zeichen;
[000]  102:
```

```
[ODL]  103:      for (k=1;k<b;k++)      /* Das 'innere'
           der Kasten mit Leerzeichen füllen */
[V30]  104:       {
[GK4]  105:        zeichen = ia + ' ';
[7M1]  106:        WARTE();
[857]  107:        ptr[(x+i)*80 + y+k] = zeichen;
[X30]  108:       }
[000]  109:
[NF6]  110:      zeichen = ra + rand[5];
[7M1]  111:      WARTE();
[P47]  112:      ptr[(x+i)*80 + y+b] = zeichen;
[X30]  113:     }
[000]  114:
[UF6]  115:    zeichen = ra + rand[6];
[7M1]  116:    WARTE();
[S77]  117:    ptr[x*80 + y+b] = zeichen;
[000]  118:
[ZF6]  119:    zeichen = ra + rand[7];
[7M1]  120:    WARTE();
[P47]  121:    ptr[(x+1)*80 + y+b] = zeichen;
[000]  122:
[76H]  123:    c_pos(x+2,y+2);                    /* Curs
           or in Kasten - Zeile 1, Spalte 1 */
[000]  124:
[BP3]  125:    return(kptr);
[X30]  126:   }
[000]  127:
[RXP]  128: /******* KA_AUS - Löschen eines Kastens, zu
           rücksetzen des Bildschirms ********
[6L8]  129: *                                          *
[8XE]  130: * liefert keinen Wert zurueck              *
[6L8]  131: *                                          *
[87B]  132: ******************************************
           *********************************/
[000]  133:
[S95]  134: void ka_aus(kptr)
[000]  135:
[RFH]  136: int far *kptr;
            /* Zeiger auf alten Bildinhalt */
[V30]  137:  {
[9NH]  138:   register int far *ptr;
                 /* Zeiger auf Bildschirm */
[ERC]  139:   int x,y,l,b;
                          /* Kasten-Daten */
[3RC]  140:   int k = 9;
                        /* Begin der Daten */
[T1F]  141:   register i,j;
                      /* Schleifenvariablen */
[000]  142:
[UVG]  143:   ptr = MK_FP(basis,0);
                /* Startadresse festlegen */
[000]  144:
[ADG]  145:   x = kptr[0] -1;
                /* Kasten-Koordinaten holen */
[BY3]  146:   y = kptr[1] -1;
[843]  147:   l = kptr[2];
[043]  148:   b = kptr[3];
[000]  149:
[NEK]  150:   for (i=0;i<=1;i++)                  /* Z
           urücksetzen des alten Bildinhalts*/
[FB4]  151:    for (j=0;j<=b;j++)
[V30]  152:     {
[7M1]  153:      WARTE();
[SN7]  154:      ptr[(x+i)*80 + y + j] = kptr[k++];
[X30]  155:     }
[000]  156:
[RZH]  157:   if(!(--anzahl))                     /* Bei
           letztem Kasten cursor wieder ein */
[F13]  158:    cur_ein();
[000]  159:
[KVK]  160:   farfree(kptr);                        /*
            Speicherbreich wieder freigeben */
[X30]  161:  }
[000]  162:
[000]  163:
[GBL]  164: /************ KA_TITEL - Funktion zum einfüg
           en eines Kasten-Titels ************
[6L8]  165: *                                          *
[UWH]  166: * liefert -1 wenn String zu lang, 0 bei Erf
           olg                                      *
[6L8]  167: *                                          *
[87B]  168: ******************************************
           *********************************/
[000]  169:
[VU7]  170: int ka_titel(kptr,sptr,ra)
[000]  171:
[8R2]  172: char *sptr;
[WW4]  173: int far *kptr, ra;
[V30]  174:  {
[SNG]  175:   register far *ptr;
                /* Pointer auf Bildschirm */
[WRF]  176:   int zeichen,
                 /* Zeichen zum schreiben */
[T3B]  177:       i,
                        /* Schleifenvar */
[MXC]  178:       laenge = 0,
                        /* Länge des Titels */
[BVC]  179:       start;
                        /* Anfangsposition */
[000]  180:
[6V6]  181:   ptr = MK_FP (get_basis(),0);
[XOE]  182:   ra <<=8;
          /* Attribut in die ersten 8 Bit */
[000]  183:
[HGG]  184:   while(*sptr++)
           /* Länge des Strings ermitteln */
[YE2]  185:    laenge++;
[000]  186:
[ED4]  187:   sptr -= laenge+1;
[28L]  188:   start = (*(kptr+3)-laenge-2)/2 + 1;
           /* Anfang im Kasten errechnen */
[ZGK]  189:   start += (kptr[0]-1)*80 + kptr[1]-1;
                /* Anfang im BS errechnen */
[000]  190:
[TUD]  191:   if (start <= 1)
                   /* Abprüfen, ob zu lang */
[6W2]  192:    return(-1);
[000]  193:
[SEF]  194:   for (i=1;i<=laenge;i++)
                        /* Begin schreiben */
[V30]  195:    {
[T96]  196:     zeichen = ra + *sptr++;
[7M1]  197:     WARTE();
[927]  198:     ptr[start+i] = zeichen;
[X30]  199:    }
[000]  200:
[ZDD]  201:   return(0);
                         /* Erfolg zurück */
[X30]  202:  }
[000]  203:
[ABG]  204: /************************ KA_LEER - Löschen
           eines Kastens ********************
[6L8]  205: *                                          *
[3LE]  206: * liefert keinen Wert zurück               *
[6L8]  207: *                                          *
[87B]  208: ******************************************
           *********************************/
[000]  209:
[L95]  210: void ka_leer(kptr)
[000]  211:
[ZH4]  212: int far *kptr;
[V30]  213:  {
[W85]  214:   register far *ptr,
[LV2]  215:            zeichen;
[852]  216:   int ia,
[FNO]  217:       i,j,
[X52]  218:       x,y,l,b;
[000]  219:
[G15]  220:   ptr = MK_FP(basis,0);
[000]  221:
[G43]  222:   x = kptr[0];
[L43]  223:   y = kptr[1];
[7N3]  224:   l = kptr[2]-1;
[ZM3]  225:   b = kptr[3]-1;
[EKG]  226:   ia = kptr[5]<<8;                    /* 'In
           nenattribut' in die ersten 8 Bit */
[000]  227:
[734]  228:   for (i=0;i<l;i++)
[D24]  229:    for (j=0;j<b;j++)
[V30]  230:     {
[GK4]  231:      zeichen = ia + ' ';
[7M1]  232:      WARTE();
[657]  233:      ptr[(x+i)*80 + y+j] = zeichen;
[X30]  234:     }
[000]  235:
[5FH]  236:   kptr[6] = x+1;                        /* K
           astencursor in Zeile 1, Spalte 1 */
[PR3]  237:   kptr[7] = y+1;
[C64]  238:   c_pos(x+1,y+1);
[X30]  239:  }
[000]  240:
[5UL]  241: /************** KA_LIES - Einen String aus
           einem Kasten einlesen *************
[6L8]  242: *                                          *
[3LE]  243: * liefert keinen Wert zurück               *
[6L8]  244: *                                          *
[87B]  245: ******************************************
           *********************************/
[000]  246:
[WX9]  247: void ka_lies(kptr,sptr,art,len)
[000]  248:
[Y65]  249: char art, *sptr;
[685]  250: int len, far *kptr;
[V30]  251:  {
[CZ1]  252:   int i=0,
[9A1]  253:       attr,
[KW1]  254:       start,
[8P1]  255:       cx,cy,
[270]  256:       l;
[UMG]  257:   register far *ptr,
                /* Pointer auf Bildschirm */
```

```
[LV2]  258:            zeichen;
[UAF]  259:    unsigned  c = ' ';
                    /* Eingegebenes Zeichen */
[000]  260:
[XHG]  261:    l = kptr[3]+kptr[0]-1;
                    /* Berechnung max. Länge */
[000]  262:
[N03]  263:    if (len < l)
[6U1]  264:     l=len;
[000]  265:
[G15]  266:    ptr = MK_FP(basis,0);
[000]  267:
[Y74]  268:    attr = kptr[5]<<8;
[000]  269:
[Y93]  270:    cx = kptr[6];
[5R3]  271:    cy = kptr[7] - 2;
[D4D]  272:    start = ((cx-1)*80) + cy;
                              /* Anfang im BS */
[000]  273:
[F13]  274:    cur_ein();
[000]  275:
[V55]  276:    while ((c != 13) && (i < l))
[V30]  277:     {
[NYD]  278:       c_pos(cx - 1,cy + i);
                              /* Cursor pos */
[P9D]  279:       c = getch();
                             /* Zeichen lesen */
[OL2]  280:       switch (c)
[V30]  281:        {
[LD9]  282:         case 27 :
                                        /* ESC */
[55B]  283:         case 13 : i = l+1;
                                   /* Return */
[KV1]  284:                   break;
[UZB]  285:         case  8 : if (i>0)
                                   /* Backspace */
[V30]  286:                   {
[4U5]  287:                    zeichen = attr + ' ';
[7M1]  288:                    WARTE();
[LH7]  289:                    ptr[start + --i] = zeichen
;
[302]  290:                    sptr--;
[X30]  291:                   }
[6PA]  292:                   else
                             /* nicht möglich */
[B23]  293:                    putch('\007');
[KV1]  294:                   break;
[BTC]  295:         default :
                            /* anders Zeichen */
[G7G]  296:         switch (toupper(art))
                         /* zulässige Zeichen */
[V30]  297:         {
[DLD]  298:          case 'A': if ((c>='A' && c<= 'Z') ||
       (c>= 'a' && c<= 'z') || c==' '
[L2B]  299:                                || c==132 || c==129 ||
       c ==148 || c==142 || c==154
[3D4]  300:                                || c==153 || c==225)
[V30]  301:                    {
[AK5]  302:                     zeichen = attr + c;
[7M1]  303:                     WARTE();
[1D7]  304:                     ptr[start + i++] = zeich
       en;
[BB3]  305:                     *sptr++ = c;
[X30]  306:                    }
[A11]  307:                   else
[B23]  308:                    putch('\007');
[KV1]  309:                   break;
[5M6]  310:          case 'X': if (c>=' ' && c<= '~'
[L2B]  311:                                || c==132 || c==129 ||
       c ==148 || c==142 || c==154
[3D4]  312:                                || c==153 || c==225)
[V30]  313:                    {
[AK5]  314:                     zeichen = attr + c;
[7M1]  315:                     WARTE();
[1D7]  316:                     ptr[start + i++] = zeich
       en;
[BB3]  317:                     *sptr++ = c;
[X30]  318:                    }
[A11]  319:                   else
[B23]  320:                    putch('\007');
[KV1]  321:                   break;
[L7D]  322:          case 'N': if ((c>= '0' && c<= '9')||
       (c=='+')||(c=='-')||(c=='/'))
[V30]  323:                    {
[AK5]  324:                     zeichen = attr + c;
[7M1]  325:                     WARTE();
[1D7]  326:                     ptr[start + i++] = zeich
       en;
[BB3]  327:                     *sptr++ = c;
[X30]  328:                    }
[A11]  329:                   else
[B23]  330:                    putch('\007');
[KV1]  331:                   break;
[UN6]  332:         }  /* ende switch (art) */
[000]  333:
[WP5]  334:       }  /* ende switch (c) */
[000]  335:
[A64]  336:     }   /* ende while */
[000]  337:
[643]  338:      cur_aus();
[703]  339:      *sptr = '\0';
```

```
[000]  340:
[X30]  341:   }
[000]  342:
[XBN]  343: /***** KA_SCHREIB - Funktion zum beschreibe
       n eines Kastens mit Strings ******
[WC8]  344: *                                          *
[UOD]  345: * liefert keinen Wert                      *
[WC8]  346: *                                          *
[7ZA]  347: ******************************************
       *********************************/
[000]  348:
[HX9]  349: void ka_schreib(kptr,str,attr)
[000]  350:
[3L5]  351: int far *kptr, attr;
[7E2]  352: char *str;
[V30]  353:  {
[W27]  354:   register int far *ptr,
[LV2]  355:            zeichen;
[803]  356:   int l,b,x,s;
[000]  357:
[G15]  358:   ptr = MK_FP(basis,0);
[000]  359:
[SR5]  360:   b = kptr[3] + kptr[1];
[000]  361:
[343]  362:   if (attr < 0)
[924]  363:    attr = kptr[5];
[762]  364:   attr <<=8;
[000]  365:
[5T3]  366:   s = kptr[7] - 1;
[000]  367:
[Y06]  368:   ptr += (kptr[6] - 1)*80 + s;
[000]  369:
[EV5]  370:   while ((s < b) && (*str))
[V30]  371:    {
[T57]  372:     zeichen = attr + *str++;
[7M1]  373:     WARTE();
[BH4]  374:     *ptr++ = zeichen;
[PH0]  375:     s++;
[X30]  376:    }
[FU3]  377:   kptr[7] = s + 1;
[000]  378:
[3XD]  379:  }
        /* ende funktion  ka_schreib */
[000]  380:
[UWH]  381: /*************** KA_CPOS - Cursor - Positio
       nierung im Kasten *****************
[6L8]  382: *                                          *
[DVN]  383: * liefert 0 bei Erfolg, -1 bei Mißerfolg (d
       .h ungültigen Werten)                 *
[6L8]  384: *                                          *
[87B]  385: ******************************************
       *********************************/
[000]  386:
[FH7]  387: int ka_cpos(kptr,cx,cy)
[000]  388:
[8W7]  389: register int far *kptr;
[US2]  390: int cx, cy;
[V30]  391:  {
[CDG]  392:   if ( (cx > (kptr[2]-1)) || (cy > (kptr[3]
       -1)) )                  /* Gültig ? */
[G03]  393:    return (-1);
[A11]  394:   else
[V30]  395:    {
[276]  396:     kptr[6] = cx + kptr[0];
[H76]  397:     kptr[7] = cy + kptr[1];
[ZF9]  398:     c_pos(cx + kptr[0],cy + kptr[1]);
[HR2]  399:     return(0);
[X30]  400:    }
[46D]  401:  }
            /* Ende Funktion ka_cpos */
[000]  402:
[SYP]  403: /*** GET_BASIS - Funktion zur Ermittlung de
       r Bildschirmspeicher - Adresse ****
[6L8]  404: *                                          *
[P2M]  405: * liefert Start-adresse des Bildschirmspeic
       hers                                  *
[6L8]  406: *                                          *
[9RN]  407: * Ermittlung erfolgt anhand der Einstellung
       der Grafikkarte                       *
[6L8]  408: *                                          *
[87B]  409: ******************************************
       *********************************/
[000]  410:
[NH6]  411: static int get_basis()
[V30]  412:  {
[GM1]  413:   int i;
[000]  414:
[8V3]  415:   for(i=0;i<99;i++)
[V30]  416:    {
[9N5]  417:     if (inport(0x3ba)==-1)
[V30]  418:      {
[ZL5]  419:       if (inport(0x3da)!=-1)
[V30]  420:        {
```

```
[R03]  421:          port = 0x3da;
[YW3]  422:          basis = 0xb800;
[F64]  423:          return(basis);
[X30]  424:         }
[X30]  425:       }
[W27]  426:      else if(inport(0x3da)==-1)
[V30]  427:       {
[H03]  428:        port = 0x3ba;
[YV3]  429:        basis = 0xb000;
[F64]  430:        return(basis);
[X30]  431:       }
[X30]  432:     }
[HR2]  433:    return(0);
[X30]  434:   }
[000]  435:
[T6H]  436: /****************** C_POS - Interrupt für
       Positionierung ********************
[CN8]  437: **
                                          **
[87B]  438: ****************************************
       **********************************/
[000]  439:
[1F6]  440: static void c_pos(x,y)
[000]  441:
[GE2]  442: int x,y;
[V30]  443:  {
[905]  444:   union REGS regs;
[000]  445:
[7KH]  446:   regs.h.ah = 2;                    /* Reg
       ister belegen für Positionierung */
[683]  447:   regs.h.bh = 0;
[EL3]  448:   regs.h.dh = x;
[RL3]  449:   regs.h.dl = y;
[000]  450:
[R7G]  451:   int86(16,&regs,&regs);
                  /* Interrupt auslösen */
[X30]  452:  }
[000]  453:
[F6B]  454: /****************************************
       **********************************
[6L8]  455: *
                                          *
[7LK]  456: * cur_aus()   - Funktion zum ausschalten de
       s Cursors                          *
[6L8]  457: *
                                          *
[87B]  458: ****************************************
       **********************************/
[000]  459:
[P26]  460: static void cur_aus()
[V30]  461:  {
[905]  462:   union REGS regs;
[000]  463:
[B83]  464:   regs.h.ah = 1;
[WL3]  465:   regs.h.ch = 32;
[0M3]  466:   regs.h.cl = 32;
[000]  467:
[DM5]  468:   int86(16,&regs,&regs);
[X30]  469:  }
[000]  470:
[F6B]  471: /****************************************
       **********************************
[6L8]  472: *
                                          *
[2KK]  473: * cur_ein()   - Funktion zum einschalten de
       s Cursors                          *
[6L8]  474: *
                                          *
[87B]  475: ****************************************
       **********************************/
[000]  476:
[L16]  477: static void cur_ein()
[V30]  478:  {
[905]  479:   union REGS regs;
[000]  480:
[B83]  481:   regs.h.ah = 1;
[493]  482:   regs.h.ch = 6;
[2M3]  483:   regs.h.cl = 14;
[000]  484:
[DM5]  485:   int86(16,&regs,&regs);
[X30]  486:  }
[000]  487:
[F6B]  488: /****************************************
       **********************************
[DNB]  489: ********************************** ENDE **
       **********************************
[87B]  490: ****************************************
       **********************************/
```

**Listing 1. In »kasten.c« sind die Funktionen zur Darstellung der Windows enthalten**

```
[F6B]    0: /****************************************
           **********************************
[6L8]    1: *
                                              *
[NMF]    2: *  KASTEN.H  - Datei muß includiert werden
                                              *
[6L8]    3: *
                                              *
[MAF]    4: *  fuer den TURBO-C - Compiler von BORLAND
           V 1.0                              *
[6L8]    5: *
                                              *
[D3D]    6: *  von Sacha Kerres, März 1988
                                              *
[6L8]    7: *
                                              *
[87B]    8: ****************************************
           **********************************/
[000]    9:
[000]   10:
[000]   11:
[OGH]   12: #define KASTEN int far *
                  /* Typ fuer Window-zeiger */
[000]   13:
[CY8]   14: extern unsigned char rand[8];
[000]   15:
[GC6]   16: extern KASTEN ka_auf();
[MV5]   17: extern void ka_aus();
[HF6]   18: extern int ka_titel();
[A96]   19: extern void ka_leer();
[H96]   20: extern void ka_lies();
[ZC7]   21: extern void ka_schreib();
[1Y5]   22: extern int ka_cpos();
```

**Listing 2. »kasten.h« müssen Sie unbedingt einbinden**

```
[ZW1]    0: test.c
[5H2]    1: kasten.c
```

**Listing 3. »test.prj«, die dazugehörende Projektdatei**

```
[8X5]    0: #include "kasten.h"
[000]    1:
[OF1]    2: main()
[V30]    3:  {
[PP6]    4:   char *str, abfrage[80];
[GM1]    5:   int i;
[AY1]    6:   KASTEN k1;
[BY1]    7:   KASTEN k2;
[000]    8:
[M05]    9:   str = "kleiner Test";
[000]   10:
[0E4]   11:   printf("\033[2J");
[LRL]   12:   printf("Dies ist eine kleine Demonstratio
          n von KASTEN.C (Taste) :");
[G12]   13:   getch();
[000]   14:
[1B8]   15:   k1 = ka_auf(10,10,10,20,4<<4|3,1<<4|5);
[000]   16:
[5N4]   17:   printf("\033[1;1H");
[B8K]   18:   printf("Und nun ein zweiter Kasten mit an
          derem Rand !                    ");
[G12]   19:   getch();
[000]   20:
[GE3]   21:   rand[0] = 192;
[RE3]   22:   rand[1] = 218;
[GF3]   23:   rand[2] = 196;
[PF3]   24:   rand[3] = 196;
[3G3]   25:   rand[4] = 179;
[9G3]   26:   rand[5] = 179;
[FF3]   27:   rand[6] = 191;
[PF3]   28:   rand[7] = 217;
[000]   29:
[0V7]   30:   k2 = ka_auf(8,12,10,40,4<<4|0,5<<4|2);
[000]   31:
[5N4]   32:   printf("\033[1;1H");
[13D]   33:   printf("Der bekommt einen Titel
                    ");
[G12]   34:   getch();
[000]   35:
[RAC]   36:   ka_titel(k2," Mein Name ist Kasten 2 ",4<
          <4|0);
[000]   37:
[5N4]   38:   printf("\033[1;1H");
```

```
[000]   39:
[784]   40:    ka_cpos(k2,2,2);
[000]   41:
[5N4]   42:    printf("\033[1;1H");
[KTB]   43:    printf("Schreiben in einen Kasten :
          ");
[G12]   44:    getch();
[000]   45:
[3V7]   46:    ka_schreib(k2,str,1<<4|0x80);
[000]   47:
[5N4]   48:    printf("\033[1;1H");
[ZK9]   49:    printf("String einlesen             ");
[000]   50:
[F84]   51:    ka_cpos(k2,4,2);
[LN8]   52:    ka_schreib(k2,"Buchstaben : ",-1);
[MA7]   53:    ka_lies(k2,abfrage,'a',10);
[R84]   54:    ka_cpos(k2,6,2);
[XN7]   55:    ka_schreib(k2,"Zahlen     : ",-1);
[NC7]   56:    ka_lies(k2,abfrage,'n',10);
[Z84]   57:    ka_cpos(k2,8,2);
[DB7]   58:    ka_schreib(k2,"Alles      : ",-1);
[7E7]   59:    ka_lies(k2,abfrage,'x',10);
[000]   60:
[000]   61:
[5N4]   62:    printf("\033[1;1H");
[BKB]   63:    printf("Löschen des Kasten-Inhalts ");
[G12]   64:    getch();
[000]   65:
[5D3]   66:    ka_leer(k2);
[000]   67:
[5N4]   68:    printf("\033[1;1H");
[90B]   69:    printf("Ausschalten der Kasten     ");
[G12]   70:    getch();
[000]   71:
[PS2]   72:    ka_aus(k2);
[000]   73:
[G12]   74:    getch();
[NS2]   75:    ka_aus(k1);
[000]   76:
[X30]   77:  }
```

**Listing 4. »test.c« ist ein kleines Demoprogramm**

```
[NH6]    0: static int get_basis()
[V30]    1:  {
[9G7]    2:   port = 0x3da;       /* oder 0x3ba  */
[WX7]    3:   basis = 0xb800;     /* oder 0xb000 */
[F64]    4:   return(basis);
[X30]    5:  }
```

**Listing 5. Wenn man nicht mit verschiedenen Grafikmodi arbeitet, läßt sich »get_b_2.c« so umschreiben**

```
[NH6]    0: static int get_basis()
[V30]    1:  {
[KL1]    2:   int c;
[000]    3:
[CE4]    4:   c = biosequip();
[9B1]    5:   c &= 48;
[000]    6:
[CX6]    7:   if (c == 48)        /* Monochrom */
[V30]    8:    {
[H03]    9:     port = 0x3ba;
[YV3]   10:     basis = 0xb000;
[X30]   11:    }
[9W4]   12:   else               /* Farbe */
[V30]   13:    {
[R03]   14:     port = 0x3da;
[YW3]   15:     basis = 0xb800;
[X30]   16:    }
[000]   17:
[F64]   18:   return(basis);
[X30]   19:  }
```

**Listing 6. Mit »get_b_2.c« können Sie ebenfalls den Modus bestimmen**


**OKI**
COMPUTERDRUCKER

**Wir bieten mehr auf PC-Basis**

**CAD-Systeme**
**Professionelle DTP-Arbeitsplätze**

**OKI Drucker Plotter**

**Standard-Software zu Superpreisen**
**Maßgeschneiderte Software**

**Spitzenservice**
**LEASING**

**Infos anfordern!**

PEACOCK COMPUTER

Computertechnik
Johannes Scheler
Postfach 1531
8058 Erding
Tel. 08122/40273

**WestComp DTP- Studio**

**Wir machen Druck !**

Sie teilen uns Ihre Wünsche mit und wir erledigen für Sie Layout, Satz und Druck mit unserer DTP-Anlage, bestehend aus:

AT-kompatiblem-Computer, Postscript- fähigem Laserdrucker, 300 dpi - Scanner

Wir bieten Ihnen ferner komplette DTP-Lösungen an. Auf Wunsch senden wir Ihnen gerne Prospektmaterial zu.

***Anfragen an:***
***WestComp GmbH***
***Helmstraße 39***
***8500 Nürnberg***

**Spitzen-Software für PCs**

**Backchart 2000** DM 295,–
Präsentationsgrafiken auf Knopfdruck – dt.)

**Brain** (Literaturdatenbank (dt.) DM 425,–

**Copy II PC** (Kopierprogramm – engl.) erhältlich auf 5.25" oder 3.5" Disks DM 79,–

**Copy II PC Deluxe Option Board** (engl.) PS DM 399,–

**Cruise Control** (Totale Kontrolle über den Cursor – engl.) DM 99,–

**Disk-Optimizer** (Harddisk-Reorganisation -engl.) DM 149,–

**FLASH** (Caching Programm – engl.) DM 265,–

**GOFER** (der "Maulwurf – speicherres. Daten-Retrieval-Prog – engl.) DM 195,–

**GRASP** (Grafik-Animations-Paket – engl. – erhältlich auf 5.25" oder 3.5" Disks) DM 389,–

**Handy-Scanner mit Texterkennung** ausführl. vorgestellt in DOS-International 8/88) DM 889,–

**INTUIT IS-2000** (integriertes Paket – engl.) DM 375,–

**Language Master Plus** (Englisch-Training – elektr. Wörterb.) DM 97,–

**Mace Vaccine** (Anti-Virus-Programm – engl.) Neu DM 79,–

**Mirror II** (Super-Kommunikations-Programm – engl.) DM 149,–

**PC-Tools de Luxe** (Utility-Paket – engl.) PS DM 149,–

**Star-Manager** (Komfortable Festplattenverwaltung mit der Maus – dt.) DM 99,–

**Star-Planer PC** Tabellenkalkulation – lotuskomp. – dt.) DM 289,–

**Star-Writer PC** (Textverarbeitung, Rechtschreibkorrektur, DFÜ, Adressverwaltung, Grafik dt.) DM 389,–

**Tornado Notes** (Formatfreie Datenbank zur Verwaltung von bis zu 25.000 Notizen – engl.) DM 275,–

**Zitate auf Diskette 1 (dt.)** Weisheiten aus drei Jahrtausenden – dt.) Neu DM 29,50

**ZyINDEX** (Programm zur Volltextrecherche – engl.)
**ZyINDEX Standard** DM 398,–
**ZyINDEX Professional** DM 775,–
**ZyINDEX Demoversion** DM 15,–

**Aus unserem Spiele-Angebot:**

**Chuck Yeager's Advanced Flight Sim.** (dt.) DM 88,–

**Interlude II** (nur für Erwachsene – engl.) DM 119,–

**King's Quest (1-3)** DM 88,–

**PeaCock XT Turbo** 640 KB – 40 MB Festpl. DM 2600,–

**Neuer Service: Computer auf Leasingbasis – wir kalkulieren Ihnen gerne eine maßgeschneiderte Lösung!**

**Software-Connection**
**Schulstr. 18 - 8913 Schondorf**
**Tel. 0 81 92/6 28**
Unser Hard- und Software-Angebot kommt auf Anfrage!

# PC - Paßwortschutz

**Haben Sie sich schon einmal darüber geärgert, daß ein lieber Arbeitskollege oder ein Familienmitglied, ohne Ihre Zustimmung, an Ihrem PC herumspielt? Wenn dem so ist, dann sollten Sie zu »Commpass« greifen.**

Viele Anwender haben schon versucht, durch ein Schutzprogramm, das automatisch beim Systemstart aufgerufen wird, ihre PCs vor unerwünschtem Zugriff zu bewahren. Diese Methode hat jedoch einen großen Nachteil: Die Datei »autoexec.bat«, die für den Aufruf des Schutzprogramms verantwortlich ist, läßt sich durch [Control-Break] abbrechen und somit wird das entsprechende Zugriffskontrollprogramm gar nicht erst aktiviert.

Hier lernen Sie eine Variante eines Paßwortschutzes für Ihr Betriebssystem kennen, die weder abgebrochen noch umgangen werden kann. Die Paßwortabfrage wird nämlich durch den Befehlsinterpreter »command.com« selbst vorgenommen. Nun brauchen Sie aber nicht zu befürchten, daß wir Ihnen das nahezu unmögliche Unterfangen zumuten wollen, einen eigenen Befehlsinterpreter zu programmieren.

Sie nehmen die Ihnen zur Verfügung stehenden Datei »command.com«, ein nicht allzu umfangreiches Assemblerprogramm, das wir Ihnen im folgenden vorstellen, ein paar Raffinessen aus der Trickkiste und schon ist Ihr eigener Command-Interpreter fertig. Sie besitzen damit einen Befehlsinterpreter, der nur dann Zugang zum Betriebssystem gestattet, wenn der Benutzer das Paßwort kennt.

Die eben erwähnten Raffinessen bestehen in der Verknüpfung des als Maschinenprogramm vorliegenden »command.com« mit einer von Ihnen selbst geschriebenen Assemblerroutine. Diese Assemblerroutine, »commpass« (Listing 1), bewirkt die Abfrage des Paßwortes und eine entsprechende Überprüfung desselben. Das zugehörige Overlay »comframe« (Listing 2) generiert die Monitormaske für die Eingabeaufforderung. Erkennt Commpass die Eingabe des richtigen Paßwortes, so wird das Betriebssystem normal gestartet. Im Falle eines falschen Paßwortes (schon beim ersten falschen Zeichen), meldet Commpass dem Benutzer, daß ihm der Zugriff verweigert wird. Daraufhin wird das Interruptsystem des Computers abgeschaltet und das System in einen Halt-Zustand versetzt.

Wie bringen Sie jetzt Ihre Routine dazu, beim Starten von »command.com« aktiviert zu werden? Nun, es ist natürlich notwendig, Ihren Befehlsinterpreter etwas abzuändern, um Commpass zu implementieren. Sie hängen die Commpass-Routine an die bestehende »command.com« an. Um dies zu realisieren, greifen Sie zu einem kleinen Trick, und zweckentfremden das Dienstprogramm Debug ein wenig. Debug benötigen Sie allerdings schon vorher; Sie müssen nämlich Ihr »command.com« vorbereiten, um die Voraussetzung für das Einbinden der Commpass-Routine zu schaffen.

Sehen Sie sich doch einmal das Listing in Bild 1 an. Hier finden Sie die durchzuführenden Schritte am Beispiel eines Befehlsinterpreters der MS-DOS Version 3.20, die zum Beispiel zum Schneider PC-1512 mitgeliefert wird, vollzogen.

Bevor Sie beginnen sollten Sie eine Diskette vorbereiten, auf die Sie die Dienstprogramme »debug«, »masm«, »link«, »exe2bin« und natürlich Ihr »command.com« kopieren. Wie Sie im Beispiel sehen, wurde zusätzlich, um Verwechslungen zu vermeiden, die Arbeitskopie des Befehlsinterpreters in »command1.com« umbenannt.

Wenn Sie nun das erste Kommando eingeben, wird Debug gestartet und »command1.com« geladen. Der Debug-Befehl »R«, zeigt Ihnen sämtliche Registerinhalte und den Befehl am momentanen Programmzeiger (IP) an. Wichtig ist hier der Inhalt von Register CX. Dort finden Sie nämlich in hexadezimaler Notation die Länge der Datei »command1.com« in Bytes (im Beispiel 5DEChex). Diesen Wert notieren Sie sich am besten gleich, da Sie ihn noch öfter benötigen werden.

Um in Ihrem Befehlsinterpreter den richtigen Einstiegspunkt für die Commpass-Routine zu finden, sind jetzt einige Überlegungen vonnöten. Der Befehlsinterpreter soll Commpass aktivieren, wenn er beim Systemstart geladen wird. Gleichzeitig muß aber gewährleistet sein, daß beim Aufruf von »»command.com«« während des normalen Betriebs, die Commpass-Routine ignoriert wird. Die ideale Gelegenheit für diese Abfrage ist die Stelle im Programmablauf, an der »command.com« die Autoexec-Datei öffnen will, um die darin abgelegten Befehle auszuführen. Die Datei »autoexec.bat« wird nämlich, ebenso wie bei Commpass gewünscht, nur beim ersten Laden des »command.com« aktiviert.

Das nächste Problem ist nun, diesen Punkt innerhalb von »command.com« zu lokalisieren. Da jeder Befehlsinterpreter genauso wie gewöhnliche Anwenderprogramme mit MS-DOS-Funktionen arbeitet, ist die Suche nicht allzu schwierig. Wenn Sie sich darauf besinnen, daß mit dem MS-DOS-Funktionsaufruf »3Dh« eine Datei geöffnet wird, deren Name in einem Speicherbereich steht, dessen Adresse wiederum in DX abgelegt sein muß, so wird das weitere Vorgehen in unserem Beispiel durchaus durchschaubar.

Mit dem Debug-Befehl »S« (Suchen) suchen Sie vom Programmbeginn (100hex) bis zum Ende des Programms (dessen Adresse sich aus der Programmlänge in CX plus dem

```
A:>DEBUG COMMAND1.COM
-R
AX=0000 BX=0000 CX=5DEC DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=43E6 ES=43E6 SS=43E6 CS=43E6 IP=0100 NV UP EI PL NZ NA PO NC
43E6:0100 E97D0C    JMP  0D80
-S100 5EEC 'AUTOEXEC.BAT'
43E6:14BE
-D14B0 L20
43E6:14B0 4D 4D 41 4E 44 2E 43 4F-4D 00 00 00 3A 5C 41 55
MMAND.COM...:\AU
43E6:14C0 54 4F 45 58 45 43 2E 42-41 54 00 FF 00 00 00 00
TOEXEC.BAT......
-S100 5EEC BA BB 14
43E6:122C
-U122C L8
43E6:122C BABB14    MOV  DX,14BB
43E6:122F B8003D    MOV  AX,3D00
43E6:1232 CD21 INT  21
-A122C
43E6:122C CALL 5EEC
43E6:122F
-W
```

**Bild 1. Die Anpassung von Commpass an das »command.com« der MS-DOS Version 3.20**

```
A>DEBUG
-N COMMPASS.COM
-L
-R
AX=0000 BX=0000 CX=600B DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=43E6 ES=43E6 SS=43E6 CS=43E6 IP=0100 NV UP EI PL NZ NA PO NC
43E6:0100 FFFF ???  DI
-N COMMAND1.COM
-L
-RCX
CX 5DEC
:600B
-W
Writing 600B bytes
-Q
```

**Bild 2. Die Verknüpfung von »command.com« und »commpass.com«**

Offset 100hex errechnet (im Beispiel 5EEChex)) nach dem Dateinamen »autoexec.bat«. Wenn Debug den gesuchten Text im Suchbereich findet, liefert es die Adresse, an der das erste Zeichen des gesuchten Textes zu finden ist (im Beispiel 14BEhex). Wenn Sie nun mit dem Debug-Befehl »D« (Dump = Anzeige) den Adreßbereich ab 14B0hex zur Anzeige bringen, sehen Sie einen Schrägstrich und einen Doppelpunkt vor dem Dateinamen. Der Schrägstrich bewirkt, daß »autoexec.bat« immer im Stammverzeichnis des aktuellen Laufwerks gesucht wird. Der Laufwerksbuchstabe wird vom Kommandointerpreter selbst ermittelt und an die Stelle vor dem Doppelpunkt abgelegt. Somit ergibt sich als reale Startadresse des Dateinamens für den Funktionsaufruf 3Dhex, nämlich die gefundene Adresse minus einem Offset von 3. Für unser Beispiel ergibt sich also eine Startadresse (Wert in »DX« beim Funktionsaufruf) von 14BBhex.

Diesen Wert notieren Sie sich, da Sie ihn bei der Commpass-Routine nochmals benötigt. Nachdem Sie nun die Adresse des Dateinamens für die Öffnung der Datei »autoexec.bat« kennen, müssen Sie noch die Stelle innerhalb des Kommandointerpreters finden, an der diese Adresse in das Register DX geladen und der Funktionsaufruf ausgeführt wird.

Das Laden des DX-Registers muß durch einen Maschinenbefehl »MOV DX,Adresse« erfolgen. Die hexadezimale Befehlssequenz hierfür lautet: »BA LL HH«, wobei »LL« das niederwertige Byte und »HH« das hochwertige Byte der Wortadresse repräsentiert. In unserem Beispiel lautet also die Sequenz zum Laden des DX-Registers mit der Adresse 14BBhex: »BA BB 14«. Mit dem Befehl »S« des Debuggers suchen Sie nun ab Adresse 100 bis zur Endadresse des Kommandointerpreters nach dieser Befehlssequenz. In unserem Beispiel wird die gesuchte Bytefolge an der Adresse 122Chex gefunden.

Mit dem Befehl »U« des Debuggers (U heißt unassemble, ist gleich deassemblieren) sehen Sie sich jetzt die Programmsequenz ab der gefundenen Adresse an. Der Wert 14BBhex (Adresse des Dateinamens »autoexec.bat«) wird in Register DX geladen, 3D00h (Funktionswert für Dateieröffnung mittels INT 21) wird in Register AX geladen und der MS-DOS-Funktionsinterrupt 21hex aufgerufen.

Da Sie Ihre Routine am Ende des Kommandointerpreters anhängen möchten, errechnet sich als Startadresse der Commpass-Routine die erste Adresse, die dem Ende von »command1.com« folgt. In unserem Beispiel ist dies 5EEChex. Sie müssen also nur dafür sorgen, daß vor dem Eröffnen der Autoexec-Datei Ihre Commpass-Routine aufgerufen wird. Sie modifizieren also mit dem »A«-Befehl (A heißt assemble, ist gleich assemblieren) des Debuggers das Kommando an der Adresse 122Chex. Der neue Befehl lautet: »CALL 5EEC« und belegt dieselbe Anzahl von Bytes wie der ursprüngliche Befehl »MOV DX,14BB«. Dadurch ist gewährleistet, daß die folgenden Anweisungen richtig interpretiert werden können.

Nun haben Sie den Kommandointerpreter vorbereitet um das Einbinden der Zusatzroutine zu ermöglichen. Mit dem Befehl »W« (W für write, ist gleich schreiben) des Debuggers wird der angepaßte Kommandointerpreter »command1.com« auf Diskette gesichert, und mit »q« verlassen Sie den Debugger.

Nun wenden Sie sich bitte dem eigentlichen Commpass-Programm (Listing 1) zu. Am Anfang des Programms finden sich zwei Parameter namens »com_length« und »aut_ex_adr«. Dem Parameter »com_length« weisen Sie bitte die aktuelle Länge Ihres Kommandointerpreters zu (in unserem Beispiel 5DEChex). Der Parameter »aut_ex_adr« erhält die ermittelte Adresse des Dateinamens »autoexec.bat« (in unserem Beispiel 14BBhex).

Nachdem Sie diese Werte angepaßt haben, geben Sie folgende Befehle ein:

```
MASM COMMPASS;
MASM COMFRAME;
LINK COMMPASS+COMFRAME,COMMPASS;
EXE2BIN COMMPASS COMMPASS.COM
```

Hierdurch wird zuerst »commpass.asm« (Listing 1) assembliert, danach die zugehörige Routine »conframe.asm« (Listing 2). Anschließend werden die erzeugten Dateien »commpass.obj« und »conframe.obj« zu »commpass.exe« verbunden und schließlich mit dem MS-DOS-Befehl »exe2bin« zu »commpass.com« konvertiert.

»commpass.com« ist das Programm, das mit dem vorbereiteten Kommandointerpreter »command1.com« verbunden werden muß. Dazu gehen Sie wie folgt vor (Bild 2):

Sie starten Debug ohne weitere Parameterangaben. Auf das Debug-Bereitschaftszeichen (-) folgend, legen Sie mit dem Befehl »N« (N steht für Name) den Namen der zu bearbeitenden Datei als »commpass.com« fest. Mit dem folgenden Befehl »L« (L für load heißt laden) wird die Datei in den Speicher eingelesen. Mit dem Befehl »R« ermitteln Sie jetzt die Länge der Datei, die in hexadezimaler Notation im Register CX zu finden ist. In unserem Beispiel hat »Commpass.com« eine Größe von 600Bhex Byte. Hierbei ist jedoch zu beachten, in dem Commpass-Programm die Größe des Kommandointerpreters durch Einfügen einer entsprechenden Anzahl von FFhex-Bytes (beginnend ab Adresse 100h) zu reservieren.

Mit dem darauffolgenden N-Befehl, weisen Sie den Debugger an, nunmehr »command1.com« als Arbeitsdatei zu betrachten. Mit dem L-Befehl laden Sie den bereits modifizierten Kommandointerpreter in den Arbeitsspeicher und überschreiben damit die als Platzhalter fungierenden FFhex Bytes des Commpass-Programms.

Es befinden sich also in diesem Moment sowohl der angepaßte Kommandointerpreter, als auch die Commpass-Routine im Speicher. Mit dem nun folgenden Debug-Befehl »RCX« verändern Sie den Inhalt des Registers DX, das im Augenblick noch die Länge der zuletzt eingelesenen Datei »command1.com« enthält, auf den Wert, der die Größe von »commpass.com« ergänzt, in unserem Beispiel also 600Bhex. Mit dem W-Befehl schreiben Sie nun die kombinierte Datei (Command1.com plus Commpass.com) unter dem Namen »command1.com« auf die Diskette.

Die Paßwortroutine ist nun in den Kommandointerpreter integriert. Fertigen Sie jetzt eine neue Startdiskette mit Ihrem Betriebssystem an, und kopieren Sie das Programm »command1.com« als »command.com« auf diese Diskette. Wenn Sie jetzt mit dieser Diskette Ihr System starten, wird der Kommandointerpreter die Eingabe eines Paßwortes verlangen. Das vorgegebene Paßwort der Commpass-Routine lautet »Test«. Geben Sie »Test« als Paßwort ein, so wird das System ganz normal geladen. Sobald jedoch ein Zeichen des Paßwortes falsch ist, stoppt Ihr Computer und läßt sich nur durch einen Kaltstart wieder zum Leben erwecken.

Das Programm »pword.com« (Listing 3) dient dazu, Ihnen die Gelegenheit zu einer komfortablen Änderung des Paßworts zu geben, es funktioniert allerdings nur bis zur MS-DOS-Version 3.2. Ab der Version 3.3 müssen Sie das Paßwort mit dem Debugger oder einem Assembler ändern.

Auch bei »pword« ist für den Parameter »com_length« zu Beginn des Programms die entsprechende Anpassung durchzuführen (Länge des ursprünglichen Kommandointerpreters angeben). Anschließend assemblieren Sie »pword«, indem Sie folgende Befehle eingeben:

```
MASM PWORD;
LINK PWORD;
EXE2BIN PWORD PWORD.COM
```

(Josef Bauer/ue)

```
[VA5]    0: title commpass.asm
[642]    1: page 66,132
[0TR]    2: ;╔═══════════════════════════════════════════════════════╗
[BZD]    3: ;|  COMMPASS.ASM          MASM COMMPASS.ASM                |
[TPE]    4: ;|                        LINK COMMPASS+COMFRAME,COMMPASS.EXE |
[Y4E]    5: ;|                        EXE2BIN COMMPASS COMMPASS.COM    |
[54A]    6: ;|                                                         |
[WSN]    7: ;|      COMMPASS wird durch DEBUG an eine zuvor modifizierte COMMAND.COM |
[11P]    8: ;|      'angehängt'. Bei einem Systemstart des PC wird somit, während |
[ABN]    9: ;|      des Ladens von COMMAND, COMMPASS aktiviert. COMMPASS erwartet die |
[U8S]   10: ;|      Eingabe eines bestimmten Passworts. Wird das korrekte Passwort ein- |
[W4R]   11: ;|      gegeben, verzweigt COMMPASS zurück zum normalen COMMAND.COM; falls |
[7KR]   12: ;|      das eingegebene Passwort nicht korrekt ist, zeigt COMMPASS eine |
[XFS]   13: ;|      Zugriffsverweigerungsmeldung an und bringt das System zum Halten. |
[5TR]   14: ;╚═══════════════════════════════════════════════════════╝
[000]   15:
[D1A]   16:          cseg segment para public 'code'
[92D]   17: assume   cs:cseg,ds:cseg,es:cseg,ss:cseg
[000]   18:
[E33]   19: public   as1
[L5K]   20: extrn    dloop:near,pos_cur:near,module1:near,exit:near
[000]   21:
[KYC]   22: com_length    equ 5DECh      ;Länge von COMMAND.COM
[D6G]   23: aut_ex_adr    equ 14BBh      ;Offsetadresse von AUTOEXEC.BAT
[000]   24:
[DL1]   25:          org 100h
[000]   26:
[KGP]   27: commpass: db com_length dup(0FFh) ;Speicherreservierung für COMMAND.COM
[000]   28:
[1M2]   29:          push ax
[7M2]   30:          push bx
[DM2]   31:          push cx
[DHF]   32:          call module1        ;COMFRAME aufrufen (Bildschirmmaske)
[5AE]   33:          mov si,offset asc1  ;Anzeigepuffer adressieren
[AEE]   34:          call dloop          ;Anzeige in der Kopfzeile liefern
[3ZC]   35:          mov dx,0C20h        ;Cursor auf Zeile 12 Spalte 32
[MXC]   36:          call dloop          ;'Password' - Anzeige liefern
[X4C]   37:          call getkey         ;Eingaberoutine aufrufen
[6LH]   38:          cmp byte ptr[check1],al ;ersten Charakter überprüfen
[RCG]   39:          jnz deny                ;bei Fehler zur Verweigerungsroutine
[643]   40:          call getkey
[TUH]   41:          cmp byte ptr[check2],al ;zweiten Charakter überprüfen
[8M2]   42:          jnz deny
[643]   43:          call getkey
[8VH]   44:          cmp byte ptr[check3],al ;dritten Charakter überprüfen
[867]   45:          jnz deny

[643]   46:          call getkey
[UUH]   47:          cmp byte ptr[check4],al ;vierten Charakter überprüfen
[8M2]   48:          jnz deny
[GBE]   49:          call exit           ;COMFRAMEs Enderoutine aktivieren
[99E]   50:          mov dx,aut_ex_adr   ;Adresse COMMAND.COMs AUTOEXEC.BAT
[B97]   51:          pop cx              ;aufsetzen
[SY1]   52:          pop bx
[LY1]   53:          pop ax
[RE9]   54:          ret                 ;Rückkehr zu COMMAND.COM
[000]   55:
[GUA]   56: deny:    mov dx,0B19h        ;Zeile 11 Spalte 25
[A8K]   57:          mov si,offset deny1 ;Verweigerungsnachricht adressieren
[LM2]   58:          push dx
[UEG]   59:          mov bx,008Fh        ;Attribut: weiß auf schwarz, blinkend
[61F]   60:          call dloop          ;Verweigerungsnachricht anzeigen
[2Z1]   61:          pop dx
[T58]   62:          inc dh              ;Zeile 12 Spalte 25
[LM2]   63:          push dx
[MU2]   64:          call dloop
[2Z1]   65:          pop dx
[Y58]   66:          inc dh              ;Zeile 13 Spalte 25
[MU2]   67:          call dloop
[SHC]   68:          cli                 ;Interruptsystem ausschalten
[939]   69:          hlt                 ;System in HALT setzen
[000]   70:
[YGH]   71: ;bei Verwendung eines NEC V20/V30 muß HLT eventuell durch
[36G]   72: ; WAIT ersetzt werden, um ein Stoppen des PC zu bewirken
[000]   73:
[B55]   74: getkey:  mov ah,0
[39D]   75:          int 16h             ;Zeichen von Tastatur einlesen
[ED2]   76:          cmp al,40h
[VL1]   77:          jl gk1
[ED2]   78:          cmp al,50h
[P8C]   79:          jge gk2             ;auf jeden Fall in einen Klein-
[2RA]   80:          or al,60h           ;buchstaben umwandeln
[G72]   81:          jmp gk1
[G44]   82: gk2:     cmp al,60h
[323]   83:          jge gk1
[422]   84:          or al,70h
[V72]   85: gk1:     ret
[000]   86:
[VY3]   87: check1:  db 't'
[DZ3]   88: check4:  db 't'
[1Z3]   89: check3:  db 's'
[7W3]   90: check2:  db 'e'
[3H7]   91: as1:     db 'Advanced Systems',0
[NZ2]   92:          db 'AS by JB',0
[AKC]   93: asc1:    db 'Processing security signon',0
[U64]   94:          db 'Password > ',0
[000]   95:
[028]   96: deny1:   db 0C9h,27 dup(0CDh),0BBh,0
[ZR9]   97: deny2:   db 0BAh,' A C C E S S   D E N I E D ',0BAh,0
[428]   98: deny3:   db 0C8h,27 dup(0CDh),0BCh,0
[000]   99:
[7M2]  100:          cseg ends
[8P3]  101:          end commpass
```

**Listing 1. Assembler-Quellcode der Paßwort-Routine »compass«**

```
[175]    0: title comframe.asm
[642]    1: page 66,132
[0TR]    2: ;╔═══════════════════════════════════════════════════════╗
[VDB]    3: ;|    COMFRAME.ASM                                       |
[Y0C]    4: ;|                        MASM COMFRAME.ASM              |
[WFE]    5: ;|                        LINK COMMPASS+COMFRAME,COMMPASS.EXE |
[E2E]    6: ;|                        EXE2BIN COMMPASS COMMPASS.COM  |
[54A]    7: ;|                                                       |
[G5N]    8: ;|          COMFRAME erstellt die Monitormaske für COMMPASS. Es schaltet |
[1UR]    9: ;|          hierzu auf Bildschirmseite 1 und sichert die aktuellen Cursor- |
[6GN]   10: ;|          Werte der Bildschirmseite 0. COMFRAME ist nur in Verbindung |
[UFE]   11: ;|          mit COMMPASS lauffähig.                      |
[5TR]   12: ;╚═══════════════════════════════════════════════════════╝
[000]   13:
[5N7]   14:          cseg segment public 'code'
[92D]   15: assume   cs:cseg,ds:cseg,es:cseg,ss:cseg
[000]   16:
[HVA]   17: public   pos_cur,dloop,exit,module1
[9H4]   18: extrn    as1:near
[000]   19:
[F2F]   20: attr     equ 014Fh      ;Bildschirmseite 1, weiß auf rot
[5TF]   21: block    equ 010Ch      ;Block-Cursor von Linie 1 bis Linie 12
[R3F]   22: hori     equ 0CDh       ;horizontales Rahmenelement '═'
[RYD]   23: vert     equ 0BAh       ;vertikales Rahmenelement '║'
[ZVE]   24: topleft  equ 0C9h       ;Rahmenelement links oben '╔'
[PFF]   25: botleft  equ 0C8h       ;Rahmenelement links unten '╚'
```

```
[V8F]   26: topright equ 0BBh       ;Rahmenelement rech
          ts oben '╗'
[NXF]   27: botright equ 0BCh       ;Rahmenelement rech
          ts unten '╝'
[4DH]   28: llink    equ 0CCh       ;linkes Rahmenverbi
          ndungselement '╠'
[AEH]   29: rlink    equ 0B9h       ;rechtes Rahmenverb
          indungselement '╣'
[2VD]   30: line     equ 78         ;Länge der horizont
          alen Linie
[000]   31:
[WG2]   32: module1:
[XV8]   33:          mov bh,0       ;Bildschirmseite 0
[1SB]   34:          mov ah,3       ;Cursor-Position er
          mitteln
[EW2]   35:          int 10h
[51K]   36:          mov word ptr[cur],dx ;Cursor-Posit
          ion der Seite 0 sichern
[SHD]   37:          mov ax,0501h   ;Bildschirmseite 1
          selektieren
[E02]   38:          int 10h
[MXA]   39:          mov cx,block   ;Block-Cursor erste
          llen
[P52]   40:          mov ah,1
[E02]   41:          int 10h
[6VC]   42:          mov ax,0600h   ;Bildschirmfenster
          erstellen
[VGA]   43:          mov bh,4Fh     ;Farbe ist weiß auf
           Rot
[C3D]   44:          xor cx,cx      ;Bildschirmecke obe
          n links ist 0,0
[82E]   45:          mov dx,1850h   ;Bildschirmecke unt
          en links ist 24,80
[EW2]   46:          int 10h
[YM8]   47:          mov bx,attr    ;Attribut laden
[2D8]   48:          mov dx,0001h   ;Zeile 0, Spalte 1
[DTA]   49:          call pos_cur   ;Cursor positionier
          en
[P9G]   50:          mov cl,line    ;Anzahl der auszuge
          benden Zeichen laden
[3W7]   51:          mov al,hori    ;Zeichen laden
[ZS7]   52:          int 10h        ;Zeile 0 anzeigen
[AYE]   53:          call skip2     ;übernächste Zeile
          positionieren
[DU7]   54:          int 10h        ;Zeile 2 anzeigen
[XX7]   55:          mov dh,22      ;Zeile 22, Spalte 1
[DTA]   56:          call pos_cur   ;Cursor positionier
          en
[PV7]   57:          int 10h        ;Zeile 22 anzeigen
[AYE]   58:          call skip2     ;übernächste Zeile
          positionieren
[PV7]   59:          int 10h        ;Zeile 24 anzeigen
[N38]   60:          xor dx,dx      ;Zeile 0, Spalte 0
[DTA]   61:          call pos_cur   ;Cursor positionier
          en
[B9F]   62:          mov cl,1       ;Anzahl der auszuge
          benden Zeichen ist 1
[KZ8]   63:          mov al,topleft ;Zeichen laden
[7S5]   64:          int 10h        ;anzeigen
[7G6]   65:          call nextline  ;Zeile 1
[UW7]   66:          mov al,vert    ;Zeichen laden
[989]   67:          push ax        ;Zeichen sichern
[7S5]   68:          int 10h        ;anzeigen
[EG6]   69:          call nextline  ;Zeile 2
[958]   70:          mov al,llink   ;Zeichen laden
[7S5]   71:          int 10h        ;anzeigen
[T3F]   72:          pop ax         ;gesichertes Zeiche
          n wiederherstellen
[L59]   73:          push ax        ;und erneut sichern
[0LB]   74:          call loop      ;Zeile 3 bis Zeile
          21 anzeigen
[EG6]   75:          call nextline  ;Zeile 22
[958]   76:          mov al,llink   ;Zeichen laden
[7S5]   77:          int 10h        ;anzeigen
[EG6]   78:          call nextline  ;Zeile 23
[T3F]   79:          pop ax         ;gesichertes Zeiche
          n wiederherstellen
[7S5]   80:          int 10h        ;anzeigen
[EG6]   81:          call nextline  ;Zeile 24
[UZ8]   82:          mov al,botleft ;Zeichen laden
[7S5]   83:          int 10h        ;anzeigen
[1L8]   84:          mov dx,004Fh   ;Zeile 0, Spalte 79
[DTA]   85:          call pos_cur   ;Cursor positionier
          en
[KY9]   86:          mov al,topright ;Zeichen laden
[7S5]   87:          int 10h        ;anzeigen
[EG6]   88:          call nextline  ;Zeile 2
[UW7]   89:          mov al,vert    ;Zeichen laden
[989]   90:          push ax        ;Zeichen sichern
[7S5]   91:          int 10h        ;anzeigen
[NG6]   92:          call nextline  ;Zeile 3
[958]   93:          mov al,rlink   ;Zeichen laden
[7S5]   94:          int 10h        ;anzeigen
[T3F]   95:          pop ax         ;gesichertes Zeiche
          n wiederherstellen
[L59]   96:          push ax        ;und erneut sichern
[CZ9]   97:          call loop      ;Zeile 3 bis 21 anz
          eigen
[EG6]   98:          call nextline  ;Zeile 22
[958]   99:          mov al,rlink   ;Zeichen laden
[7S5]  100:          int 10h        ;anzeigen
[EG6]  101:          call nextline  ;Zeile 23
[T3F]  102:          pop ax         ;gesichertes Zeiche
          n wiederherstellen
[7S5]  103:          int 10h        ;anzeigen
[EG6]  104:          call nextline  ;Zeile 24
[UY9]  105:          mov al,botright ;Zeichen laden
[7S5]  106:          int 10h        ;anzeigen
[ZC8]  107:          mov dx,1703h   ;Zeile 23, Spalte 3
[SYC]  108:          mov si,offset as1 ;Anzeigetext adr
          essieren
[SP7]  109:          call dloop     ;Text anzeigen
[H48]  110:          mov dl,69      ;Zeile 23, Spalte 6
          9
[VK9]  111:          call dloop     ;Folgetext anzeigen
[KD8]  112:          mov dx,0103h   ;Zeile 1, Spalte 3
[DTA]  113:          call pos_cur   ;Cursor positionier
          en
[YM8]  114:          mov bx,attr    ;Attribut laden
[TZ9]  115:          ret            ;Rückkehr zum Modul
           COMMPASS
[000]  116:
[N3B]  117: exit:    mov bh,0       ;Bildschirmseite 0
[TXC]  118:          mov ax,0500h   ;Bildschirmseite 0
          auswählen
[E02]  119:          int 10h
[55G]  120:          mov dx,word ptr[cur] ;gesicherte C
          ursor-Position
[AV8]  121:          mov ah,2       ;wiederherstellen
[E02]  122:          int 10h
[TZ9]  123:          ret            ;Rückkehr zum Modul
           COMMPASS
[000]  124:
[K84]  125: skip2:   inc dh
[PD5]  126: nextline:inc dh
[8C5]  127: pos_cur: push ax
[7M2]  128:          push bx
[DM2]  129:          push cx
[G2B]  130:          mov ah,02h     ;Cursor-Position se
          tzen
[E02]  131:          int 10h
[XY1]  132:          pop cx
[SY1]  133:          pop bx
[L52]  134:          pop ax
[2PC]  135:          mov ah,09h     ;Zeichen und Attrib
          ut anzeigen
[SM0]  136:          ret
[000]  137:
[HXA]  138: loop:    mov cx,19      ;Anzahl der Zeilen
[NXA]  139: l1:      call nextline  ;nächste Zeile
[EN3]  140:          call pos_cur
[29C]  141:          push cx        ;Schleifenzähler si
          chern
[VVF]  142:          mov cl,1       ;Anzahl der anzuzei
          genden Zeichen laden
[UK8]  143:          int 10h        ;Zeichen anzeigen
[HPE]  144:          pop cx         ;Schleifenzähler wi
          ederherstellen
[P62]  145:          loop l1
[VVF]  146:          mov cl,1       ;Anzahl der anzuzei
          genden Zeichen laden
[SM0]  147:          ret
[000]  148:
[8YB]  149: dloop:   mov bx,attr    ;Attribut laden
[EN3]  150:          call pos_cur
[9E8]  151:          lodsb          ;Textzeichen laden
[4ND]  152:          or al,al       ;prüfen, ob Textend
          e erreicht ist
[CM2]  153:          jz enddl
[3YF]  154:          mov cx,1       ;Anzahl der anzuzei
          genden Zeichen laden
[UK8]  155:          int 10h        ;Zeichen anzeigen
[K49]  156:          inc dl         ;Spaltenwert erhöhe
          n
[9S2]  157:          jmp dloop
[NV1]  158: enddl:
[SM0]  159:          ret
[000]  160:
[8C2]  161: cur:     dw 0
[000]  162:
[7M2]  163:          cseg ends
[D13]  164:          end module1
```

**Listing 2. Assembler-Quellcode der Eingabemaske »comframe«**

```
[475]    0: title pword.asm
[642]    1: page 66,132
[0TR]    2: ;╔═══════════════════════════════════════
                ═══════════════════════════════════╗
[YKD]    3: ;║   PWORD.ASM              MASM PWORD.ASM
                                                   ╟
[B2C]    4: ;║                          LINK PWORD.OBJ
```

```
                                        ║
[ZZC]  ;║                       EXE2BIN PWORD PWOR
       D.COM                            ║
[54A]  ;║                               ║
                                        ║
[S6P]  ;║           Dieses Dienstprogramm ermög
       licht dem Anwender die Vergabe   ║
[Z5P]  ;║           eines selbstgewählten Passw
       ortes für einen durch COMMPASS   ║
[0EE]  ;║           erweiterten COMMAND.COM.
                                        ║
[5TR]  ;╚═══════════════════════════════════
       ═════════════════════════════════╝
[000]
[98A]            pwordseg segment public 'code'
[59L]  assume    cs:pwordseg,ds:pwordseg,es:pwordse
       g,ss:pwordseg
[000]
[YVD]  com_length    equ 5DECh          ;Länge vo
       n COMMAND.COM
[G3H]  pw_offset     equ com_length+74h ;Offset d
       es Passworts in
[0R2]                                   ;COMMAND.
       COM
[DL1]            org 100h
[000]
[5N5]  pword:    call signon
[Z3E]            call get_attr          ;Dateiatt
       ribut ermitteln
[B0B]            call open_file         ;COMMAND.
       COM öffnen
[2TB]            call read_file         ;COMMAND.
       COM einlesen
[TKD]            call close_file        ;COMMAND.
       COM schließen
[3BB]            call pw_disp           ;Passwort
        anzeigen
[TDC]            call pw_change         ;Passwort
        editieren
[B0B]            call open_file         ;COMMAND.
       COM öffnen
[H4D]            call write_file        ;COMMAND.
       COM schreiben
[UWG]            call set_attr          ;Dateiatt
       ribut wiederherstellen
[Y4A]            jmp term               ;Programm
        beenden
[000]
[HK6]  get_attr: mov ax,4300h
[825]            mov dx,offset file
[BGC]            int 21h                ;Dateiatt
       ribut ermitteln
[TNB]            mov word ptr [f_attr],cx ;und sich
       ern
[CWE]            jc open_fail           ;bei Fehl
       er zur Fehlerroutine
[WY2]            mov ax,4301h
[825]            mov dx,offset file
[0H2]            mov cx,20h
[6XB]            int 21h                ;Dateiatt
       ribut auf 'ARCHIV'
[LAF]            jc write_fail          ;bei Fehl
       er zur Fehlerroutine
[SM0]            ret
[000]
[2L6]  set_attr: mov ax,4301h
[825]            mov dx,offset file
[H6G]            mov cx,word ptr [f_attr] ;Original
       -Dateiattribut
[RR9]            int 21h                ;wiederhe
       rstellen
[LAF]            jc write_fail          ;bei Fehl
       er zur Fehlerroutine
[SM0]            ret
[000]
[EG6]  open_file: mov ax,03D02h
[825]            mov dx,offset file
[AU9]            int 21h                ;COMMAND.
       COM eröffnen
[1WG]            mov word ptr [handle],ax ;Bearbeit
       ungsnummer sichern
[CWE]            jc open_fail           ;bei Fehl
       er zur Fehlerroutine
[SM0]            ret
[000]
[6Z2]  read_file:
[B72]            mov bp,0
[X8G]            mov bx,word ptr [handle] ;Bearbeit
       ungsnummer lesen
[1NH]            mov dx,offset file_buf   ;Adresse
       des Lesepuffers setzen
[SZ9]            mov cx,25000           ;Dateilän
       ge setzen
[CE2]            mov ah,3Fh
[1U9]            int 21h                ;COMMAND.
       COM einlesen
[3SE]            jc read_fail           ;bei Fehl
       er zur Fehlerroutine
[NEK]            mov word ptr [file_len],ax ;aktuel
       le Dateilaenge sichern
```

```
[SM0]            ret
[000]
[7A3]  write_file:
[PY2]            mov ax,4000h
[X8G]            mov bx,word ptr [handle] ;Bearbeit
       ungsnummer lesen
[RZF]            mov dx,offset file_buf   ;Adresse
       des Schreibpuffers
[SFK]            mov cx,word ptr [file_len] ;Aktuel
       le Dateilänge ermitteln
[YHA]            int 21h                ;COMMAND.
       COM schreiben
[LAF]            jc write_fail          ;bei Fehl
       er zur Fehlerroutine
[SM0]            ret
[000]
[MZ6]  close_file: mov ah,3Eh
[X8G]            mov bx,word ptr [handle] ;Bearbeit
       ungsnummer lesen
[95B]            int 21h                ;COMMAND.
       COM schließen
[SM0]            ret
[000]
[WX6]  term:     call close_file
[LT9]            mov ah,4Ch             ;Programm
        beenden
[T02]            int 21h
[000]
[HKA]  signon:   mov dx,offset sign_msg
[YV4]            mov ah,9h
[T02]            int 21h
[SM0]            ret
[000]
[H4B]  open_fail: mov dx,offset o_fail_msg
[NKC]            mov ah,9h              ;Dateierö
       ffnungsfehler
[YGE]            int 21h                ;anzeigen
        und Programm beenden
[MP2]            jmp term
[000]
[G2B]  read_fail: mov dx,offset r_fail_msg
[FSB]            mov ah,9h              ;Lesefehl
       er anzeigen und
[3F9]            int 21h                ;Programm
        beenden
[MP2]            jmp term
[000]
[8YB]  write_fail: mov dx,offset w_fail_msg
[2CD]            mov ah,9h              ;Schreibf
       ehler anzeigen und
[3F9]            int 21h                ;Programm
        beenden
[MP2]            jmp term
[000]
[KUA]  pw_disp:  mov si,offset file_buf
[B25]            add si,pw_offset
[VVG]            mov di,offset old_pw   ;Bisherig
       es Passwort ermitteln
[XDA]            movsb                  ;und in P
       uffer lesen
[Y52]            add si,2
[3L1]            movsb
[C72]            sub si,2
[3L1]            movsb
[C72]            sub si,2
[3L1]            movsb
[KT7]            mov dx,offset old_pword
[KMD]            mov ah,9h              ;Bisherig
       es Passwort anzeigen
[T02]            int 21h
[SM0]            ret
[000]
[TLA]  pw_change: mov dx,offset ch_msg
[Y82]            mov ah,9h
[T02]            int 21h
[LLD]            call get_pw            ;neues Pa
       sswort erbitten
[YT7]            mov dx,offset new_pword
[2UC]            mov ah,9h              ;neues Pa
       sswort anzeigen
[T02]            int 21h
[4CE]            mov dx,offset ok_msg   ;Richtigk
       eit des neuen
[VTA]            int 21h                ;Passwort
       es nochmals
[D0A]            mov ah,1               ;bestätig
       en lassen
[T02]            int 21h
[4VB]            or al,70h              ;Groß/Kle
       inschreibung
[BA8]            cmp al,'y'             ;überbrüc
       ken
[G64]            jnz pw_change
[85E]            call conv_low          ;Groß/Kle
       in-Konvertierung
[W68]            mov si,offset new_pw

[0U6]            mov di,offset file_buf
[005]            add di,pw_offset
[AHD]            movsb                  ;neues Pa
```

```
          sswort in COMMAND.COM
[E17]  141:           add di,2                    ;einfügen
[3L1]  142:           movsb
[152]  143:           sub di,2
[3L1]  144:           movsb
[152]  145:           sub di,2
[3L1]  146:           movsb
[LD6]  147:           mov dx,offset upd_msg
[V1C]  148:           mov ah,9h                   ;erfolgre
          iche Änderung
[AV6]  149:           int 21h                     ;anzeigen
[SM0]  150:           ret
[000]  151:
[SR5]  152: get_check: mov ah,9h
[T02]  153:           int 21h
[FLC]  154: get_pw:   mov ah,1                    ;Eingabef
          unktion
[CAD]  155:           mov cx,4                    ;Länge de
          r Eingabe festlegen
[HU5]  156:           mov di,offset new_pw
[Y4D]  157: get_pw_loop: int 21h                  ;Zeichen
          einlesen
[LTB]  158:           stosb                       ;und in P
          uffer schreiben
[0RC]  159:           cmp al,21h                  ;Kontroll
          zeichenfilter
[C9C]  160:           jb get_check                ;wenn ja,
           wiederholen
[Z3F]  161:           loop get_pw_loop            ;nächstes
           Zeichen einlesen
[SA1]  162:           ret
[000]  163:
[FHA]  164: conv_low: mov si,offset new_pw
[N2E]  165:           mov di,si                   ;Eingabe
          wird in jedem Fall in
[ZKE]  166:           mov cx,4                    ;Kleinsch
          reibung umgewandelt, da
[BEF]  167: conv_loop: lodsb                      ;COMMPASS
           ein Passwort in
[6SC]  168:           cmp al,40h                  ;Kleinbuc
          hstaben erwartet
[311]  169:           jl c1
[ED2]  170:           cmp al,50h
```

```
[2G1]  171:           jge c2
[X12]  172:           or al,60h
[9H1]  173:           jmp c1
[AH5]  174: c2:       cmp al,60h
[WF1]  175:           jge c1
[422]  176:           or al,70h
[W63]  177: c1:       stosb
[HU4]  178:           loop conv_loop
[SM0]  179:           ret
[000]  180:
[H5C]  181: ok_msg      db 'Is the new PASSWORD ok (y or
          n)? $'
[B1C]  182: ch_msg      db 0DH,0AH,0AH,'Enter new PASSWO
          RD: ','$'
[VHA]  183: old_pword   db 'Current PASSWORD is -> '
[C85]  184: old_pw      db 'OOOO','$'
[K4C]  185: new_pword   db 0DH,0AH,'New PASSWORD will be
          -> '
[UF7]  186: new_pw      db 'NNNN',0DH,0AH,0AH,'$'
[4XG]  187: sign_msg    db 'Copyright (c) Advanced Syste
          ms - AS by JB - 1987'
[A33]  188:             db 0DH,0AH,0AH
[Y6E]  189:             db 'Advanced Systems',27h,' PASS
          WORD editor',0DH,0AH,0AH,'$'
[5DG]  190: o_fail_msg db 'Cannot open the file \COMMAN
          D.COM !!!',0DH,0AH,0AH,'$'
[5DG]  191: r_fail_msg db 'Cannot read the file \COMMAN
          D.COM !!!',0DH,0AH,0AH,'$'
[2AF]  192: w_fail_msg db 'Cannot write to \COMMAND.COM
           !!!',0DH,0AH,0AH,'$'
[7C5]  193: upd_msg     db 0DH,0AH
[HHE]  194:             db 'Updating \COMMAND.COM with n
          ew PASSWORD finished.'
[FG2]  195:             db 0DH,0AH,'$'
[U44]  196: f_attr      dw 0
[H04]  197: handle      dw 0
[9F4]  198: file_len    dw 0
[C56]  199: file        db '\COMMAND.COM',0
[V47]  200: file_buf    db 'Buffer.......'
[000]  201:
[ZE4]  202:           pwordseg ends
[MU2]  203:           end pword
```

**Listing 3. Zum Paßwort definieren: »pword**


# Datenaustausch über die serielle Schnittstelle für IBM und kompatible PC's

Ob Laptop, XT, AT, ob 3 1/2" oder 5 1/4" Laufwerk, Festplatte oder Ramdisk...

PC-LINK schafft die Möglichkeit, daß zwei über die serielle Schnittstelle verbundene Rechner alle vorhandenen Diskettenlaufwerke, Festplatten und Drucker gemeinsam benutzen können.

Die Verbindung schafft das Kabel (im Preis enthalten) – eine äußerst schnelle 115200-baud-Verbindung. Einfach anstecken am RS 232-Stecker (=V24).

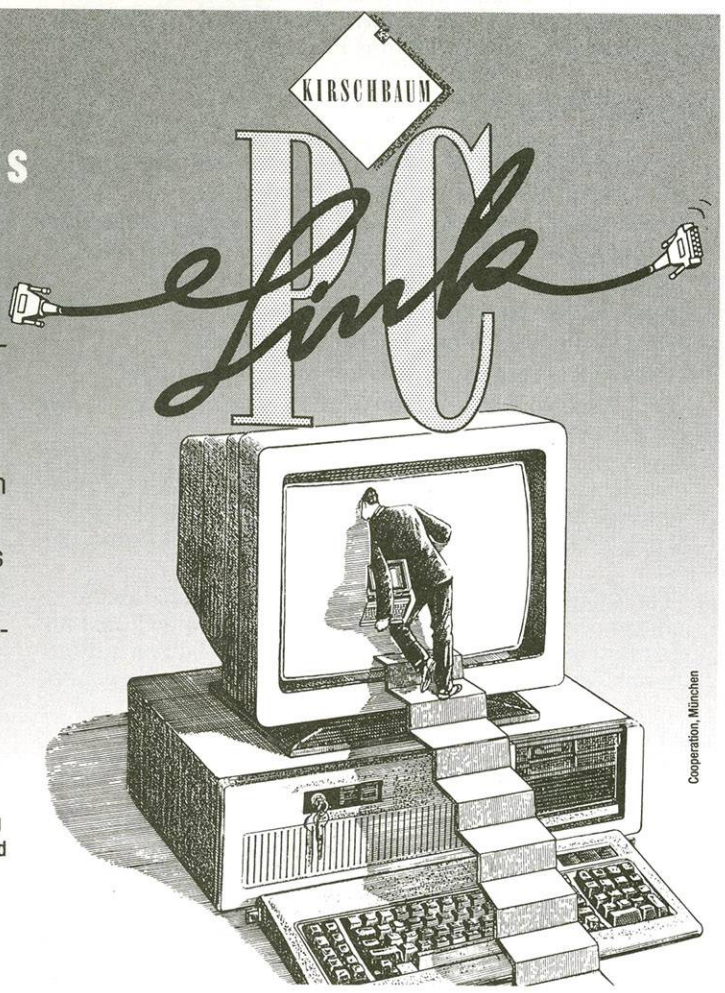
Daten werden absolut fehlerfrei übertragen, das garantiert das von PC-LINK benutzte Übertragungsprotokoll.

Die Installierung von PC-LINK ist in wenigen Sekunden geschehen – ohne Fachkenntnisse!

Ein Beispiel: Vom Laptop mit 3 1/2" Diskettenlaufwerk sollen Dateien auf einen AT mit 5 1/4" Laufwerk übertragen werden. Der Befehl lautet: COPY A:*.* D:.

Kirschbaum PC-LINK erhalten Sie im Fachhandel. Unverbindliche Preisempfehlung DM 168,– (Im Lieferumfang enthalten: Deutsches Handbuch, Verbindungskabel und Software auf 5 1/4" – und 3 1/2")

KIRSCHBAUM SOFTWARE GMBH
Kronau 15 · 8091 Emmering · Tel. 08067/1220 · Fax 08067/1053



# Kommando-Editor für MS-DOS

**Wer auf Betriebssystemebene ein falsches Zeichen eingegeben hat, muß mit der Rücktaste sämtliche Eingaben wieder löschen und den Befehl nochmals schreiben. Mit »expacomm.com« können Sie ab jetzt ihre Eingabe bequem editieren, und die acht letzten Befehle werden auf Tastendruck bereitgehalten. Aber das Utility kann noch mehr...**

Mit diesem speicherresidenten Hilfsprogramm, das Sie am besten in Ihre »autoexec.bat«-Datei einfügen, stehen Ihnen leistungsfähige Funktionen zur Verfügung. »expacomm.com« ist in Assembler geschrieben und deshalb nur etwas über 4900 Byte groß. »expacomm.com« bietet Ihnen folgende Funktionen:

- einen Kommandoeditor, der ein komfortables Editieren der Eingabezeile bietet
- einen Befehlsstack, der die letzten acht Eingabezeilen speichert und die Befehlszeilen auf Tastendruck ausgibt
- eine Funktionstastenbelegung, die es erlaubt, die Befehle in einer Datei zu speichern oder bereits gespeicherte Eingaben wieder auf die Funktionstasten zu übertragen.

Das Programm kann eigenen Bedürfnissen angepaßt werden. So ist es beispielsweise kein Problem, mehr als acht Eingaben zwischenzuspeichern oder den Namen der Funktionstastendatei zu ändern. Hier die Erläuterung der verschiedenen Fähigkeiten des Programms.

### Der Kommandoeditor

- [Pfeil rechts] bewegt den Cursor eine Spalte nach rechts, löscht aber keine Zeichen. Am Zeilenende kann der Cursor nicht weiterbewegt werden.
- [Pfeil links] bewegt den Cursor in der Eingabezeile eine Spalte nach links, Zeichen werden nicht gelöscht.
- [Home] bringt den Cursor an den Zeilenanfang.
- [End] bringt den Cursor ans Zeilenende.
- [Del] löscht das Zeichen unter dem Cursor.
- [Backspace] löscht das Zeichen links vom Cursor.
- [Esc] löscht die ganze Zeile und der Cursor steht wieder am Zeilenanfang.
- [Ins] schaltet den Einfügemodus ein, und man kann an der Cursorposition ein Zeichen einfügen. Der Einfügemodus schaltet sich nach Eingabe der Tasten [CR], [Esc], [Pfeil hoch], [Pfeil tief] oder einer Funktionstaste ab.
- [Ret] übergibt eine Zeile an »command.com«. »Expacomm.com« gibt aber nur den Teil links vom Cursor an das Betriebssystem ab. Befindet sich der Cursor mitten in der Zeile, muß man mit [End] ans Zeilenende gehen.

### Der Befehlsstack

»expacomm.com« installiert einen Stack für die letzten acht Eingaben.

- mit [Pfeil hoch] wird der Inhalt des Stacks beginnend mit dem ersten Eintrag in Richtung des letzten Eintrags durchlaufen. Jeder Eintrag wird in die Eingabezeile kopiert und Sie können mit [Ret] den Befehl bestätigen oder vorher noch editieren.
- mit [Pfeil tief] wird der Inhalt des Stacks in umgekehrter Reihenfolge durchlaufen.

Der Stack ist als Ringpuffer angelegt, so daß nach acht Eingaben der verfügbare Platz ausgeschöpft ist und ein neuntes Kommando wieder den ersten Befehl überschreibt.

### Die Funktionstastenbelegung

Mit der Syntax »eco fx text« können Sie die Funktionstasten mit Eingaben versehen. Der Befehl »eco« teilt »expacomm.com« mit, daß die Funktionstasten belegt werden. Für »x« geben Sie die Ziffer der Funktionstaste ein, dabei wird »F10« mit »F0« angesprochen. »Text« steht für den MS-DOS-Befehl.

Wenn Sie beispielsweise »eco f0 dir a:*.asm« eingeben, erscheint auf dem Bildschirm, nachdem Sie die Taste [F10] gedrückt haben, die Meldung »dir a:*.asm«. Geben Sie den Befehl »eco f7 date« ein, dann gibt [F7] auf dem Bildschirm »date« aus.

Mit dem Befehl »eco w« kann die Funktionstastenbelegung als Datei mit dem Namen »f__tasten.eco« auf dem aktuellen Laufwerk gespeichert werden. »eco r« stattet die Funktionstasten mit den Befehlen der Datei »f__tasten.eco« wieder aus. Existiert die Datei nicht, erfolgt eine Fehlermeldung.

Um den Inhalt der zehn Funktionstasten auf dem Bildschirm zu bekommen, drücken Sie die Taste [PgUp].

»eco ?« aktiviert ein Hilfsmenü, das die Kommandos kurz erklärt.

»eco #« deaktiviert »expacomm.com«. Ein erneutes Starten von »expacomm.com« ist möglich.

»eco t« zeigt die Tiefe des Befehlsstacks an.

### Die Programmstruktur

Das Programm »expacomm.com« ist dem MS-DOS-Befehlsprozessor vorgeschaltet und fängt dadurch alle Eingaben ab. Erst nach der Bearbeitung werden die Befehle an »command.com« weitergegeben.

Alle Eingaben an die »Gepufferte Tastatureingabe« gehen grundsätzlich über die Funktion 0Ahex des MS-DOS-Interrupts 21hex, der alle Befehle, die über die Tastatur eingegeben werden, in einen Puffer ablegt. Die Eingaben werden gepuffert, bis die Return-Taste gedrückt wird. Dies geschieht folgendermaßen:

1) Die Prozessorregister DS und DX werden mit der Pufferanfangsadresse geladen.

2) Das erste Byte des Puffers muß die maximale Anzahl der entgegenzunehmenden Zeichen enthalten, einschließlich Returnzeichen. Dieses Byte ist also mit dem erforderlichen Wert geladen. Das zweite Pufferbyte wird von MS-DOS benutzt. Hier steht nach der Ausführung des Interrupts die Anzahl der eingetippten Zeichen ohne dem Returnzeichen. Ab dem dritten Byte steht die eigentliche Tastatureingabe.

3) Das Register AH wird mit dem Funktionscode 0Ahex für die »Gepufferte Tastatureingabe« geladen und anschließend der Interrupt 21hex ausgelöst.

## Der Kommandoeditor setzt sich aus verschiedenen Komponenten zusammen

Nach der Ausführung der Interruptroutine folgt die Auswertung des Eingabepuffers durch den Befehlsprozessor. Doch an dieser Stelle wird »expacomm.com« aktiv, denn bei der Initialisierung von »expacomm.com« wird ein neuer Interruptvektor 21hex installiert. Dieser neue Vektor zeigt jetzt auf einen Programmteil von »expacomm.com« und nicht mehr auf die ursprüngliche Betriebssystemroutine. Der Programmteil trägt im Listing den Namen »newint«. Hier wird untersucht, ob die Funktion 0Ahex ausgeführt werden soll. Ist das der Fall, übernimmt »expa-

comm.com« diese Aufgabe, andernfalls wird die Programmausführung an der Stelle fortgesetzt, auf die der alte Interruptvektor zeigt.

Dadurch besitzt »expacomm.com« die volle Eingabekontrolle. Die Eingaben kann man jetzt beliebig manipulieren, bevor sie in den Puffer kopiert werden, den der Befehlsprozessor für die Tastatureingabe vorgesehen hat. Die Adresse des Puffers läßt sich aus den Registerinhalten von DS und DX ablesen, die für die Ausführung des Interrupts mit dieser Adresse geladen werden mußten. Die Adresse wird für das Zurückkopieren später benötigt und deshalb gespeichert. Die nun folgende Programmschleife »warte« liest alle Tastatureingaben ein und legt sie in einem Puffer ab. Drückt man eine Funktionstaste, werden die entsprechenden Unterprogramme aufgerufen. Die Schleife wird durch »return« verlassen.

Durch einen Druck auf die Return-Taste durchsucht »expacomm.com« den Puffer um festzustellen, ob die Eingabe Expacomm-Kommandos enthält. Ist das nicht der Fall, wird die Zeile in den Eingabepuffer des Befehlsprozessors kopiert und gleichzeitig im Befehlsstack aufgenommen. Der Befehlsprozessor übernimmt wieder die Kontrolle und führt den Befehl aus. Ist aber in der Eingabe ein Kommando für »expacomm.com« enthalten, wird in die entsprechenden Routinen des Programms verzweigt und der Befehl ausgeführt. Dem Befehlsprozessor täuscht man eine leere Eingabe vor, und es findet auch kein Eintrag im Befehlsstack statt.

Diese Strukturbeschreibung für den Ablauf des Programms sollte ausreichen. Einen genaueren Überblick bietet das Listing, das ausreichend mit Kommentaren versehen ist.

Hier noch einige Hinweise, um »expacomm.com« den eigenen Wünschen anzupassen. Ein Erhöhen der Stacktiefe erreicht man durch das Ändern der Konstante »max_cmdz_len« im Quelltext.

Da »expacomm.com« ein Editieren über eine Bildschirmzeile hinaus nicht gestattet, sollte der Prompt nicht zu lang sein. Wer aber im Prompt auf Datum und Uhrzeit nicht verzichten möchte, kann die Definition seines Prompts mit einem Zeilenvorschub am Ende versehen. Das Programm ist darauf ausgelegt, einen Platz von mindestens 60 Zeichen auf der Eingabezeile vorzufinden, also sollte die Promptlänge in dieser Zeile nicht mehr als 20 Zeichen betragen.

Das Kommandoschlüsselwort »eco« läßt sich den eigenen Vorstellungen entsprechend anpassen. Dazu ist lediglich im Datenbereich des Quelltextes die Vereinbarung von »cmdkeyword« zu ändern.

Der Zeichenbereich, den die Eingaberoutine als gültig akzeptiert, kann man durch das Ändern der Konstanten »valid_code_min« oder »valid_code_max« einschränken oder vergrößern. Die Definitionen im Quelltext erlauben nur Eingaben zwischen 20hex und FFhex. Werte, die außerhalb des Bereichs liegen, lehnt »expacomm.com« mit einem Piepston ab.

(Harald Rieckmann/gh)

```
[VL6]    0: ;*****************************************
            ***
[SAC]    1: ;** harald rieckmann, braunschweig 1/1988 *
            ***
[VL6]    2: ;*****************************************
            ***
[V10]    3: ;
[642]    4: page 66,132
[MWF]    5: msdos            equ     21h
            ; dos interrupt function call
[LLD]    6: video            equ     10h
            ; bios video interrupt
[V10]    7: ;
[1WH]    8: drive_number     equ     0h
            ; offset zu drive number im fcb.
[5VG]    9: record_size      equ     0eh
            ; offset zu record size im fcb.
[0MK]   10: cur_record       equ     20h
            ; offset zu current record im fcb.
[V10]   11: ;
[BOT]   12: ; um ein einwandfreies funktionieren von ex
            pacomm zu gewaehrleisten, darf die
[M9M]   13: ; eingabezeilenlaenge nicht laenger als (79
            - promptlaenge) sein!!!
[K9E]   14: cmdz_len        equ     60
            ; eingabezeilenlaenge.
[V10]   15: ;
[ZCK]   16: max_cmdz_puffer equ     8
            ; max stackgroesse in zeilen.
[R9C]   17: cr              equ     0dh
            ; einige tastencodes.
[8V3]   18: lf              equ     0ah
[724]   19: esc             equ     1bh
[BK4]   20: space           equ     20h
[2M5]   21: backspace       equ     08h
[PSK]   22: valid_code_min  equ     20h
            ; zeichenbereich, der angenommen
[RXA]   23: valid_code_max  equ     0ffh
            ; wird.
[V10]   24: ;
[18H]   25: f1scan          equ     3bh
            ; scan code der funktionstasten.
[Z05]   26: f2scan          equ     3ch
[315]   27: f3scan          equ     3dh
[715]   28: f4scan          equ     3eh
[B15]   29: f5scan          equ     3fh
[2Y4]   30: f6scan          equ     40h
[6Y4]   31: f7scan          equ     41h
[AY4]   32: f8scan          equ     42h
[EY4]   33: f9scan          equ     43h
```

```
[N85]   34: f10scan         equ     44h
[TG5]   35: homescan        equ     47h
[N25]   36: upscan          equ     48h
[TK5]   37: pgupscan        equ     49h
[6M5]   38: leftscan        equ     4bh
[CT5]   39: rightscan       equ     4dh
[EM5]   40: endscan         equ     4fh
[AK5]   41: downscan        equ     50h
[FG5]   42: pgdnscan        equ     51h
[SK5]   43: insscan         equ     52h
[EH5]   44: delscan         equ     53h
[V10]   45: ;
[LD5]   46: cseg            segment
[9NA]   47:                 assume  cs:cseg,ds:cseg,es:
        cseg
[V10]   48: ;
[54L]   49: ; initialisiere und setze den neuen interru
        ptvektor fuer
[BBF]   50: ; int 21h funktion 0Ah: buffered keyboard i
        nput.
[SRG]   51: ; anschliessend wird das programm resident
        gemacht.
[V10]   52: ;
[902]   53:                 org     100h
[B76]   54: expacomm:       jmp     init
[V10]   55: ;
[PNG]   56: ;=========================================
        ====================================
[9EF]   57: ;               unterprogramme fuer den neue
        n command handler
[PNG]   58: ;=========================================
        ====================================
[V10]   59: ;
[V10]   60: ;
[L2H]   61: copy_zeile_stack proc near ;---------------
        ------------------------------------
[PUF]   62:                  ; eine eingabezeile wird i
        n den stack aufgenommen.
[V10]   63:                 ;
[RN2]   64:                 xor     ch,ch
[M2H]   65:                 mov     cl,cmdz.aktlen
            ; anzahl zu kopierender zeichen.
[W08]   66:                 add     cx,2
            ; + aktlen. + cr.
[D38]   67:                 mov     si,offset cmdz.aktl
        en
[M8G]   68:                 mov     di,cmd_stack_ptr_e
            ; ziel und quelle einstellen.
[BG6]   69:                 rep movsb
            ; copy.
```

```
[X57]   70:                     mov     cx,cmd_stack_ptr_e
[TGD]   71:                     add     cx,cmdz_len +1
          ; erhoehe stack_ptr oder
[OCF]   72:                     cmp     cx,offset cmd_stack
        _end    ; bei ueberlauf ...
[Y14]   73:                     jb      inc_ptr
[UYD]   74:                     mov     cx,offset cmd_stack
          ; ... beginne von vorn.
[L9B]   75: inc_ptr:            mov     cmd_stack_ptr_e,cx
[7D7]   76:                     mov     cmd_stack_ptr_a,cx
[SMO]   77:                     ret
[KGG]   78: copy_zeile_stack endp ;---------------------
        -----------------------------------
[V10]   79: ;
[66G]   80: copy_to_zeile       proc near ;-----------------
        -----------------------------------
[M1M]   81:                     ; aus dem f-tastenpuffer od
          er dem stack wird eine zeile in den
[5UK]   82:                     ; kommandozeilenpuffer kopi
          ert. in si muss die quelle stehen.
[V10]   83:                     ;
[CY9]   84:                     mov     di,offset cmdz.aktl
          en ; ziel.
[XP9]   85:                     mov     cl,[si]
          ; zeilenlaenge.
[CY1]   86:                     cmp     cl,0
[BYF]   87:                     je      ctz_aus
          ; keinen leeren eintrag behandeln.
[7YD]   88:                     mov     cmdz_ptr,cl
          ; cmdz.ptr ans zeilenende.
[RN2]   89:                     xor     ch,ch
[PBC]   90:                     add     cx,2
          ; zeilenlaenge +aktlen  +cr.
[BG6]   91:                     rep movsb
          ; copy.
[TK4]   92:                     call    display
[NG4]   93: ctz_aus:            ret
[RDF]   94: copy_to_zeile       endp ;---------------------
        -----------------------------------
[V10]   95: ;
[F3H]   96: copy_zeile_comm proc near ;-----------------
        -----------------------------------
[LKM]   97:                     ; aus dem eingabezeilenpuff
          er wird die eingegebene zeile in den
[TFK]   98:                     ; puffer des aufrufenden co
          mmand.com-files uebertragen.
[V10]   99:                     ;
[KFC]  100:                     push    es
          ; achtung! anderes es-segment.
[LXH]  101:                     mov     cl,cmdz.aktlen
          ; anzahl der zu kopierenden bytes.
[RN2]  102:                     xor     ch,ch
[MN9]  103:                     add     cx,3
          ;  +maxlen +aktlen +cr.
[L79]  104:                     mov     si,offset cmdz
          ; quelle.
[3MG]  105:                     mov     ax,command_ds
          ; segment vom command.com puffer.
[SS2]  106:                     mov     es,ax
[OHG]  107:                     mov     di,command_dx
          ; offset vom command.com puffer.
[BG6]  108:                     rep movsb
          ; copy.
[9N1]  109:                     pop     es
[SMO]  110:                     ret
[OBG]  111: copy_zeile_comm endp ;---------------------
        -----------------------------------
[V10]  112: ;
[WOF]  113: neubeginn           proc near ;-----------------
        -----------------------------------
[K5M]  114:                     ; cursor springt an den anf
          ang der kommandozeile zurueck und
[5CC]  115:                     ; es wird die zeile geloesc
          ht. neubeginn.
[V10]  116:                     ;
[86C]  117:                     mov     insflag,0
          ; einfuegemodus aus.
[KCC]  118:                     call    homekey
          ; an den zeilenanfang.
[9XG]  119:                     mov     cl,cmdz.aktlen
          ; anzahl der zu loeschen zeichen.
[RN2]  120:                     xor     ch,ch
[PGD]  121:                     cmp     cx,0
          ; falls zeile leer, loeschen un-
[N57]  122:                     je      nb_aus
          ; noetig.
[114]  123:                     mov     al,space
[UNH]  124: nb_weiter:          mov     ah,0eh
          ; zeile mit spaces loeschen.
[3Z1]  125:                     mov     bh,0
[MW2]  126:                     int     video
[FP4]  127:                     loop    nb_weiter
[DEF]  128:                     call    homekey
          ; wieder zum anfang zurueck.
[ZLF]  129: nb_aus:             mov     cmdz.aktlen,0
          ; zeilenlaenge ist 0.
```

```
[SMO]  130:                     ret
[D8E]  131: neubeginn           endp ;---------------------
        -----------------------------------
[V10]  132: ;
[RPF]  133: clear_screen        proc near ;----------------
        -----------------------------------
[6Y7]  134:                     ; loescht den bildschirm.
[14K]  135:                     ; setzt cursorposition auf
          obere linke bildschirmecke.
[EYH]  136:                     ; in bh muss die seite steh
          en, die angesprochen werden soll.
[V10]  137:                     ;
[5Z1]  138:                     push    bx
[5WD]  139:                     mov     ax,0600h
          ; scroll window. die ganze seite.
[CG2]  140:                     mov     cx,0h
[692]  141:                     mov     dl,79
[H72]  142:                     mov     dh,24
[H23]  143:                     mov     bh,0fh
[MW2]  144:                     int     video
[GN1]  145:                     pop     bx
[DG2]  146:                     mov     dx,0h
[F68]  147:                     mov     ah,02h
          ; cursor home.
[MW2]  148:                     int     video
[SMO]  149:                     ret
[TCE]  150: clear_screen        endp ;---------------------
        -----------------------------------
[V10]  151: ;
[ZWE]  152: del_left            proc near ;----------------
        -----------------------------------
[HZC]  153:                     ; loescht das zeichen links
          vom cursor.
[V10]  154:                     ;
[3HC]  155:                     cmp     cmdz_ptr,0
          ; nicht am zeilenanfang.
[A04]  156:                     je      del_aus
[K3C]  157:                     dec     cmdz_ptr
          ; zeiger nach links.
[43A]  158:                     dec     curpos_x
          ; update curpos.
[MRF]  159:                     call    set_cur
          ; cursor nach links, dann delkey
[6U3]  160:                     call    delkey
[EE4]  161: del_aus:            ret
[G4E]  162: del_left            endp ;---------------------
        -----------------------------------
[V10]  163: ;
[NPE]  164: display             proc    near ;-------------
        -----------------------------------
[UAL]  165:                     ; stellt die zeile, die aus
          dem stack oder dem f-tastenpuffer
[V8C]  166:                     ; in die kommandozeile kopi
          ert wird, dar.
[V10]  167:                     ;
[P9H]  168:                     mov     cl,cmdz.aktlen
          ; anzahl darzustellender zeichen.
[RN2]  169:                     xor     ch,ch
[R1E]  170:                     add     curpos_x,cl
          ; curpos_x + zeilenlaenge.
[U67]  171:                     mov     bx,offset cmdz.text
[K36]  172: displ:              mov     ah,0eh
[KPE]  173:                     mov     al,[bx]
          ; hole zeichen einzeln aus cmdz
[X5A]  174:                     int     video
            ; und zeige sie an.
[OM1]  175:                     inc     bx
[283]  176:                     loop    displ
[AZ1]  177:                     mov     ah,2
[ULF]  178:                     mov     dx,curpos
          ; neue curpos auf dem bildschirm
[496]  179:                     mov     bh,0
          ; setzen.
[MW2]  180:                     int     video
[1T4]  181: no_display:         ret
[YAE]  182: display             endp ;---------------------
        -----------------------------------
[V10]  183: ;
[NNE]  184: set_cur             proc    near ;-------------
        -----------------------------------
[87G]  185:                     mov     dx,curpos
          ; setze den bildschirmcursor auf
[HEC]  186:                     mov     ah,2
          ; stelle, die in curpos steht.
[BZ1]  187:                     mov     bl,0
[MW2]  188:                     int     video
[SMO]  189:                     ret
[Y9E]  190: set_cur             endp ;---------------------
        -----------------------------------
[V10]  191: ;
[57F]  192: funkt_edit          proc near ;----------------
        -----------------------------------
[MGG]  193:                     ; in ah steht der scan code
          der funktionstaste, al = 0.
[T7K]  194:                     ; ueber die tabelle werden
          alle routinen angesprungen.
```

```
[V1O]  195:                    ;
[MS8]  196:                    cmp     ah,f1scan
       ; f1-taste?
[8C3]  197:                    jb      return
[8O8]  198:                    cmp     ah,delscan
       ; delkey?
[1OE]  199:                    ja      return
       ; wenn nicht dazwischen, dann
[HK7]  200:                    sub     ah,f1scan
       ; return.
[NOE]  201:                    xchg    ah,al
       ; in al steht die ordnungszahl
[4Z1]  202:                    push    ax
[MPC]  203:                    shl     al,1
       ; mal 2, tabelleneintraege sind
[W2A]  204:                    lea     di,funkttable
       ; 2 bytes lang.
[3NC]  205:                    add     di,ax
       ; die f-tasten von 0 fuer f1
[7K9]  206:                    pop     ax
       ; bis 24 fuer delkey
[T4H]  207:                    jmp     cs:word ptr [di]
       ; ueber die tabelle anspringen.
[V1O]  208:
       ;
[93F]  209: funkttable         label   byte
       ; einsprungtabelle
[TN7]  210:                    dw      offset f1_taste
       ; 3bh
[UN7]  211:                    dw      offset f2_taste
       ; 3ch
[VN7]  212:                    dw      offset f3_taste
       ; 3dh
[WN7]  213:                    dw      offset f4_taste
       ; 3eh
[XN7]  214:                    dw      offset f5_taste
       ; 3fh
[5P7]  215:                    dw      offset f6_taste
       ; 40h
[6P7]  216:                    dw      offset f7_taste
       ; 41h
[7P7]  217:                    dw      offset f8_taste
       ; 42h
[8P7]  218:                    dw      offset f9_taste
       ; 43h
[SL7]  219:                    dw      offset f10_taste
       ; 44h
[P65]  220:                    dw      offset return
[P65]  221:                    dw      offset return
[A87]  222:                    dw      offset homekey
       ; 47h
[GE6]  223:                    dw      offset upkey
       ; 48h
[L97]  224:                    dw      offset pgupkey
       ; 49h
[P65]  225:                    dw      offset return
[Z87]  226:                    dw      offset leftkey
       ; 4bh
[P65]  227:                    dw      offset return
[UT7]  228:                    dw      offset rightkey
       ; 4dh
[P65]  229:                    dw      offset return
[ZT6]  230:                    dw      offset endkey
       ; 4fh
[OA7]  231:                    dw      offset downkey
       ; 50h
[K87]  232:                    dw      offset pgdnkey
       ; 51h
[4V6]  233:                    dw      offset inskey
       ; 52h
[DU6]  234:                    dw      offset delkey
       ; 53h
[V1O]  235:                    ;
[BOC]  236: ;-----------------------------------------
       -----------------------------------
[CCG]  237: return:            ; dummy fuer fehlenden tabe
       lleneintrag
[V1O]  238:                    ;
[SMO]  239:                    ret
[BOC]  240: ;-----------------------------------------
       -----------------------------------
[FOL]  241:                    ; je nach eigegebener f-tas
       te, wird der entsprechende eintrag
[74A]  242:                    ; in die kommandozeile kopi
       ert.
[V1O]  243:                    ;
[WFP]  244: f10_taste:         ; sonderbehandlung fuer f10
       , da spez. anordnung des puffers.
[M93]  245:                    lea     si,ft10
[TG3]  246:                    jmp     source
[KX2]  247: f1_taste:
[MX2]  248: f2_taste:
[PX2]  249: f3_taste:
[SX2]  250: f4_taste:
[UX2]  251: f5_taste:
[WX2]  252: f6_taste:
```

```
[YX2]  253: f7_taste:
[OY2]  254: f8_taste:
[2Y2]  255: f9_taste:
[ZCC]  256:                    mov     ah,0
       ; in al steht 0-8 je nach taste.
[9ND]  257:                    mov     bl,cmdz_len +1
       ; laenge der pufferzeile.
[DLC]  258:                    mul     bl
       ; berechne adresse des puffers
[XOD]  259:                    lea     si,ft1
       ; fuer die jeweilige taste.
[UN2]  260:                    add     si,ax
[3O8]  261: source:            call    neubeginn
[8AH]  262:                    call    copy_to_zeile
       ; hier muss si uebergeben werden.
[SMO]  263:                    ret
[BOC]  264: ;-----------------------------------------
       -----------------------------------
[Z5L]  265: homekey:           ; bewegt den cursor an den
       anfang der eingabezeile
[V1O]  266:                    ;
[LT4]  267:                    mov     al,cmdz_ptr
[5HH]  268:                    sub     curpos_x,al
       ; berechne schritte bis zum anfang.
[FWC]  269:                    mov     cmdz_ptr,0
       ; setze zeiger zurueck.
[U1A]  270:                    call    set_cur
       ; setze cursor
[SMO]  271:                    ret
[BOC]  272: ;-----------------------------------------
       -----------------------------------
[GOG]  273: upkey:             ; holt eine zeile aus dem s
       tack zurueck
[V1O]  274:                    ;
[8H4]  275:                    call    neubeginn
[757]  276:                    mov     si,cmd_stack_ptr_a
[TZF]  277:                    call    copy_to_zeile
       ; holt eine zeile vom stack.
[N57]  278:                    mov     cx,cmd_stack_ptr_a
[TGD]  279:                    add     cx,cmdz_len +1
       ; erhoehe stack_ptr oder
[OCF]  280:                    cmp     cx,offset cmd_stack
       _end    ; bei ueberlauf ...
[D64]  281:                    jb      up_inc_ptr
[UYD]  282:                    mov     cx,offset cmd_stack
       ; ... beginne von vorn.
[MCB]  283: up_inc_ptr:        mov     cmd_stack_ptr_a,cx
[SMO]  284:                    ret
[6AC]  285: ;-----------------------------------------
       -----------------------------------
[G5H]  286: pgupkey:           ; zeigt auf dem bildschirm
       die aktuelle
[5B7]  287:                    ; f-tasten-belegung an.
[V1O]  288:                    ;
[8H4]  289:                    call    neubeginn
[PDH]  290:                    call    clear_screen
       ; schirm loeschen und cursor home.
[NC7]  291:                    mov     dx,offset f_tabelle
[602]  292:                    mov     ah,9
[7UB]  293:                    int     msdos
       ; schreibe tabellenkopf.
[172]  294:                    mov     dl,10
[9B2]  295:                    mov     dh,"0"
[FU9]  296:                    lea     si,ft10
       ; tabellenanfang.
[KG6]  297: next_ft:           xor     ch,ch
[702]  298:                    mov     bx,1
[OM7]  299:                    mov     ax,0e46h
       ; display 'F'
[76E]  300:                    int     video
       ; vorsicht, ah wird zerstoert!
[A72]  301:                    mov     ah,14
[YPC]  302:                    mov     al,dh
       ; display laufenden index.
[MW2]  303:                    int     video
[7Z1]  304:                    push    dx
[BK8]  305:                    mov     dx,offset doppelpun
       kt
[602]  306:                    mov     ah,9
[NW7]  307:                    int     msdos
       ; display ': '
[KN1]  308:                    pop     dx
[KS6]  309:                    mov     cl,byte ptr [si]
[Z28]  310:                    cmp     cl,0
       ; kein eintrag.
[142]  311:                    je      leer
[N3B]  312: displ_ft:          mov     al, byte ptr [si+bx
       ]
[A72]  313:                    mov     ah,14
[14C]  314:                    int     video
       ; display zeile bis ende.
[OM1]  315:                    inc     bx
[6G4]  316:                    loop    displ_ft
[W7G]  317: leer:              add     si,cmdz_len +1
       ; naechste f-tastenzeile.
[RE3]  318:                    mov     ax,0e0dh
```

```
[MW2]  319:                    int     video
[CG6]  320:                    mov     ax,0e0ah
           ; cr,lf.
[MW2]  321:                    int     video
[C86]  322:                    inc     dh
           ; index + 1.
[LK1]  323:                    dec     dl
[AWB]  324:                    jne     next_ft
           ; alle f-tasten durch?
[FN1]  325:                    pop     ax
[G95]  326:                    jmp     war_kommando
[BOC]  327: ;-------------------------------------
           ---------------------------------
[DEN]  328: leftkey:           ; der cursor bewegt sich ei
           nen schritt nach links ueber
[WDK]  329:                    ; die eingabezeile. ueberst
           richene buchstaben werden nicht
[4G3]  330:                    ; geloescht.
[V10]  331:                    ;
[HE4]  332:                    cmp     cmdz_ptr,0
[Y2C]  333:                    je      left_aus
           ; nicht am zeilenanfang!
[VM9]  334:                    dec     curpos_x
           ; cursor links.
[F84]  335:                    dec     cmdz_ptr
[LSF]  336:                    call    set_cur
           ; setze neue cursor position.
[YD4]  337: left_aus:          ret
[BOC]  338: ;-------------------------------------
           ---------------------------------
[2CC]  339: rightkey:          ; entsprechend leftkey (s.o
           .)
[V10]  340:                    ;
[LT4]  341:                    mov     al,cmdz_ptr
[N96]  342:                    cmp     al,cmdz.aktlen
[CHB]  343:                    je      right_aus
           ; nicht am zeilenende!
[02A]  344:                    inc     curpos_x
           ; cursor rechts.
[694]  345:                    inc     cmdz_ptr
[LSF]  346:                    call    set_cur
           ; setze neue cursor position.
[XE4]  347: right_aus:         ret
[BOC]  348: ;-------------------------------------
           ---------------------------------
[ATK]  349: endkey:            ; bewegt den cursor zum end
           e der eingabenzeile
[V10]  350:                    ;
[NA6]  351:                    mov     al,cmdz.aktlen
[9SE]  352:                    sub     al,cmdz_ptr
           ; berechne die schritte zum
[WZ9]  353:                    add     curpos_x,al
           ; zeilenende.
[3BB]  354:                    add     cmdz_ptr,al
           ; zeiger ans ende.
[ORF]  355:                    call    set_cur
           ; setze neue cursorposition.
[SMO]  356:                    ret
[BOC]  357: ;-------------------------------------
           ---------------------------------
[GVG]  358: downkey:           ; holt eine zeile aus dem s
           tack zurueck
[V10]  359:                    ;
[N57]  360:                    mov     cx,cmd_stack_ptr_a
[PHF]  361:                    sub     cx,cmdz_len +1
           ; vermindere stack_ptr oder
[30D]  362:                    cmp     cx,offset cmd_stack
           ; bei ueberlauf ...
[L65]  363:                    jae     down_dec_ptr
[MTE]  364:                    lea     cx,last_stack_entry
           ; ... beginne von hinten.
[11C]  365: down_dec_ptr:      mov     cmd_stack_ptr_a,cx
[8H4]  366:                    call    neubeginn
[757]  367:                    mov     si,cmd_stack_ptr_a
[TZF]  368:                    call    copy_to_zeile
           ; holt eine zeile vom stack.
[ZH4]  369: downaus:           ret
[6AC]  370: ;-------------------------------------
           ---------------------------------
[1X3]  371: pgdnkey:           ;
[SMO]  372:                    ret
[6AC]  373: ;-------------------------------------
           ---------------------------------
[OHP]  374: inskey:            ; schaltet den einfuegemodu
           s ein. (wird nach jedem cr und esc
[T18]  375:                    ; wieder ausgeschaltet).
[V10]  376:                    ;
[455]  377:                    mov     insflag,0ffh
[SMO]  378:                    ret
[6AC]  379: ;-------------------------------------
           ---------------------------------
[CBP]  380: delkey:            ; loescht das zeichen auf d
           em sich der cursor gerade befindet.
[V10]  381:                    ;
[5Z1]  382:                    push    bx
```

```
[MT4]  383:                    mov     bl,cmdz_ptr
[TU5]  384:                    cmp     cmdz.aktlen,bl
[AWA]  385:                    jle     d_aus
           ;i nicht am zeilenende.
[KN2]  386:                    xor     bh,bh
[25K]  387: del1:              mov     al,cmdz.text[bx+1]
           ; von rechts aufruecken...
[FE6]  388:                    mov     cmdz.text[bx],al
[OM1]  389:                    inc     bx
[P96]  390:                    cmp     bl,cmdz.aktlen
[EW1]  391:                    je      del2
[008]  392:                    mov     ah,0eh
           ;    display
[MW2]  393:                    int     video
[AXC]  394:                    jmp     del1
           ; ...bis zeilenende erreicht
[XC6]  395: del2:              mov     al,space
[9ZC]  396:                    mov     ah,0eh
           ; letztes zeichen loeschen
[MW2]  397:                    int     video
[WXE]  398:                    dec     cmdz.aktlen
           ; zeilenlaenge korrigieren.
[46A]  399:                    call    set_cur
           ; setze cursor.
[4P4]  400: d_aus:             pop     bx
[SMO]  401:                    ret
[V10]  402: ;
[PEE]  403: funkt_edit         endp ;---------------------
           ---------------------------------
[V10]  404: ;
[Y9D]  405: peep               proc    near ;-------------
           ---------------------------------
[ZW9]  406:                    ; es wird ein piepston erze
           ugt.
[V10]  407:                    ;
[3N9]  408:                    mov     ax,0e07h
           ; ctrl-g ausgeben.
[MW2]  409:                    int     video
[SMO]  410:                    ret
[7YC]  411: peep               endp ;---------------------
           ---------------------------------
[V10]  412: ;
[U1G]  413: zeichenablegen     proc    near ;-------------
           ---------------------------------
[DVK]  414:                    ; von der tastatur eingegeb
           ene zeichen (al) werden je nach mo-
[Z8M]  415:                    ; dus mit oder ohne einfueg
           en in die kommandozeile aufgenommen.
[V10]  416:                    ;
[4RG]  417:                    cmp     byte ptr insflag,0f
           fh   ; mit oder ohne einfuegen?
[AG5]  418:                    je      mit_einfuegen
[8UE]  419: ohne_einfuegen:   xor     bh,bh
           ; ohne einfuegen:
[1U8]  420:                    mov     bl,cmdz_ptr
           ;----------------
[LAG]  421:                    mov     cmdz.text[bx],al
           ; in den puffer ablegen und auf
[4SC]  422:                    mov     ah,0eh
           ; dem bildschirm anzeigen.
[MW2]  423:                    int     video
[2XB]  424:                    inc     curpos_x
           ; update cursorpos.
[97H]  425:                    cmp     cmdz.aktlen,bl
           ; innerhalb eines wortes? ja, dann
[GZD]  426:                    ja      ab1
           ; keine aenderung von cmdz.aktlen.
[9ZE]  427:                    inc     cmdz.aktlen
           ; sonst erhoehe zeilenlaenge.
[ROE]  428: ab1:               inc     cmdz_ptr

           ; erhoehe zeilen-ptr.
[3VB]  429:                    jmp     ab_aus
           ; ende ohne einfuegen.
[V10]  430:                    ;
[HRG]  431: mit_einfuegen:     mov     bl,cmdz.aktlen
           ; mit einfuegen:
[LG8]  432:                    cmp     bl,cmdz_ptr
           ;----------------
[X5K]  433:                    je      ohne_einfuegen
           ; am zeilenende einfuegen unnoetig.
[KN2]  434:                    xor     bh,bh
[YTE]  435:                    lea     si,cmdz.text[bx]
           ; si weist auf das letzte
[06E]  436:                    mov     di,si
           ; di hinter das letzte zeichen.
[4L1]  437:                    inc     di
[4T2]  438:                    mov     cx,bx
[DZE]  439:                    sub     cl,cmdz_ptr
           ; so viele zeichen verschieben,
[5GB]  440:                    inc     cl
           ; aber cr nicht vergessen!
[6Z1]  441:                    push    cx
[HY7]  442:                    std
           ; rueckwaerts!!!
```

```
[BM8]  443:                    rep     movsb
                ; verschiebe!
[L2E]  444:                    cld
                ; unbedingt wieder zuruecknehmen!
[MT4]  445:                    mov     bl,cmdz_ptr
[HMG]  446:                    mov     cmdz.text[bx],al
                ; einzufuegendes zeichen in die
[6B9]  447:                    pop     cx
                ; zeile eintragen.
[YU6]  448: anzeigen:          mov     ah,0eh
[MW2]  449:                    int     video
[68F]  450:                    inc     bx
                ; zeichen von einfuegeposition bis
[2LE]  451:                    mov     al,cmdz.text[bx]
                ; zeilenende anzeigen.
[GG4]  452:                    loop    anzeigen
[694]  453:                    inc     cmdz_ptr
[784]  454:                    inc     curpos_x
[SG4]  455:                    inc     cmdz.aktlen
[HXE]  456:                    call    set_cur
                ; bildschirmcursor auf die ein-
[D2E]  457: ab_aus:            ret
                ; fuegestelle zurueck.
[3PF]  458: zeichenablegen  endp ;----------------------
               -------------------------------------
[V1O]  459: ;
[DLE]  460: kommando?          proc    near ;-----------
               -------------------------------------
[23L]  461:                    ; es wird ueberprueft, ob i
               n der kommandozeile eine anweisung
[GLL]  462:                    ; enthalten ist. ist das de
               r fall, so wird sie ausgefuehrt und
[Y5E]  463:                    ; mit gesetztem carryflag z
               urueckgekehrt.
[23C]  464:                    ; anderfalls wird das carry
                geloescht.
[V1O]  465:                    ;
[227]  466:                    mov     cx,cmdkeyword_len
[N1B]  467:                    lea     si,cmdz.text
                ; kommandozeile.
[WS2]  468:                    mov     bx,si
[1BG]  469:                    lea     di,cmdkeyword
                ; steht in der kommandozeile das
[AMB]  470:
                ; kommandoschluesselwort (default:
[H4F]  471:                    repe cmpsb
                ; eco)?, wenn nicht, dann zurueck.
[DC2]  472:                    jne     kaus
[9Z6]  473:                    add     bx,cmdkeyword_len
[WEF]  474:                    mov     al,[bx]
                ; bx zeigt auf das 1. zeichen nach
[TTF]  475:                    cmp     al,space
                ; dem kommandowort. hier muss ein
[R9D]  476:                    jne     kaus
                ; space stehen, sonst fehler!
[UP3]  477:                    mov     al,[bx+1]
[Z3E]  478:                    cmp     al,"f"
                ; als naechstes muss ein zeichen
[VSC]  479:                    jne     k1
                ; folgen, das den befehl benennt.
[Y95]  480:                    jmp     prog_ftasten
[LK4]  481: k1:                cmp     al,"r"
[4G1]  482:                    jne     k2
[664]  483:                    jmp     read_file
[EL4]  484: k2:                cmp     al,"w"
[6G1]  485:                    jne     k3
[DF4]  486:                    jmp     write_file
[KOD]  487: k3:                cmp     al,"t"
                ; springe zum kommando.
[8G1]  488:                    jne     k4
[ZW4]  489:                    jmp     stapeltiefe
[UA4]  490: k4:                cmp     al,"?"
[AG1]  491:                    jne     k5
[5U4]  492:                    jmp     help_screen
[G64]  493: k5:                cmp     al,"#"
[DC2]  494:                    jne     kaus
[5V4]  495:                    jmp     disactivate
[5GK]  496: kaus:              jmp     kein_kommando
                ; war nicht dabei oder fehler.
[V1O]  497: ;
[UGE]  498: prog_ftasten:      ;-------------------------
               -------------------------------------
[PUA]  499:
                ; hinter dem befehlszeichen muss
[UDD]  500:                    cmp     byte ptr [bx+3],spa
               ce  ; ein space folgen.
[CE3]  501:                    jne     falsch
[UP3]  502:                    mov     al,[bx+2]
[D6D]  503:                    cmp     al,"0"
                ; gueltiger bereich fuer die
[G6A]  504:                    jb      falsch
                ; funktionstaste?
[VB2]  505:                    cmp     al,"9"
[N54]  506:                    jbe     richtig
[369]  507: falsch:            jmp     kein_kommando
```

```
[NKK]  508: richtig:           sub     al,"0"
                ; berechnung der ft_pufferadresse
[H9B]  509:                    mov     ah,0
                ; der einzelnen f-tasten.
[SZC]  510:                    mov     cl,cmdz_len +1
                ; pufferzeilenlaenge.
[PM1]  511:                    mul     cl
[KS2]  512:                    mov     si,ax
[1W3]  513:                    lea     di,ft10[si]
[9Y1]  514:                    push    di
[4L1]  515:                    inc     di
[R86]  516:                    lea     si,[bx+4]
                ;    <4->
[LU4]  517:
                ; eco f1 dir a:<cr>
[3CH]  518:                    mov     cl,cmdz.aktlen
                ;  bx^ si^  (hierhin zeigen bx, si)
[S2G]  519:                    xor     ch,ch
                ; berechnung der zu verschiebenden
[X6D]  520:                    sub     cx,cmdkeyword_len
                ; zeichen : minus "eco"
[6EA]  521:                    sub     cx,3
                ; minus "_fx_" + cr: -4+1=-3
[6Z1]  522:                    push    cx
[NAB]  523:                    rep     movsb
                ; copy in den ft-puffer.
[HN1]  524:                    pop     cx
[MM1]  525:                    pop     di
[KK1]  526:                    dec     cl
[STC]  527:                    mov     [di],cl
                ; aktlen in den ft-puffer.
[MSE]  528:                    stc
                ; cf=1, melde: kommado ausgefuehrt.
[CY4]  529:                    jmp     ausgefuehrt
[V1O]  530: ;
[UKD]  531: read_file:         ; -------------------------
               -------------------------------------
[52A]  532:                    mov     dx,offset fcb
                ; open file.
[0B7]  533:                    mov     fcb[drive_number],0
[G23]  534:                    mov     ah,0fh
[VY2]  535:                    int     msdos
[4S3]  536:                    cmp     al,0ffh
[GN8]  537:                    je      r2
                ; can't open file.
[2Z1]  538:                    mov     ah,0
[PZD]  539:                    mov     fcb[cur_record],ah
                ; start mit record 0.
[2C5]  540:                    mov     al,cmdz_len +1
[1ZG]  541:                    mov     word ptr fcb[record
               _size],ax ; record size setzen.
[7HE]  542:                    lea     si,ft10
                ; disk transfer address ist ft10.
[L3A]  543:                    mov     cx,10
                ; 10 mal records lesen.
[OR4]  544: r1:                mov     dx,si
[V13]  545:                    mov     ah,1ah
[HD8]  546:                    int     msdos
                ; setze dta.
[E9A]  547:                    mov     dx,offset fcb
                ; read record.
[UU2]  548:                    mov     ah,14h
[VY2]  549:                    int     msdos
[AY1]  550:                    cmp     al,0
[VW2]  551:                    jne     r_aus
[GYG]  552:                    add     si,cmdz_len +1
                ; neue dta ist naechster ft-puffer.
[BV1]  553:                    loop    r1
[WOD]  554: r_aus:             mov     dx,offset fcb
                ; close file.
[6U2]  555:                    mov     ah,10h
[VY2]  556:                    int     msdos
[673]  557:                    jmp     r_aus2
[AY3]  558: r2:                mov     ah,9
[SCG]  559:                    mov     dx,offset no_file
                ; meldung: file nicht gefunden.
[VY2]  560:                    int     msdos
[CT3]  561: r_aus2:            stc
[CY4]  562:                    jmp     ausgefuehrt
[V1O]  563: ;
[VUD]  564: write_file:        ; -------------------------
               -------------------------------------
[WZE]  565:                    mov     fcb[drive_number],0
                ; current drive benutzen.
[VOA]  566:                    mov     dx,offset fcb
                ; create file.
[4V2]  567:                    mov     ah,16h
[VY2]  568:                    int     msdos
[4S3]  569:                    cmp     al,0ffh
[ZHA]  570:                    je      w_aus
                ; can't create file.
[DN2]  571:                    xor     ah,ah
[PZD]  572:                    mov     fcb[cur_record],ah
                ; start mit record 0.
[2C5]  573:                    mov     al,cmdz_len +1
```

```
[1ZG]  574:                    mov     word ptr fcb[record
       _size],ax ; record size setzen.
[529]  575:                    lea     si,ft10
       ; 1. dta ist ft10.
[5GB]  576:                    mov     cx,10
       ; 10 mal record schreiben.
[5R4]  577: w1:                mov     dx,si
[V13]  578:                    mov     ah,1ah
[VY2]  579:                    int     msdos
[XFA]  580:                    mov     dx,offset fcb
       ; write record.
[ZU2]  581:                    mov     ah,15h
[VY2]  582:                    int     msdos
[AY1]  583:                    cmp     al,0
[OX2]  584:                    jne     w_aus
[GYG]  585:                    add     si,cmdz_len +1
       ; neue dta ist naechster ft-puffer.
[GV1]  586:                    loop    w1
[11D]  587: w_aus:             mov     dx,offset fcb
       ; close file.
[6U2]  588:                    mov     ah,10h
[VY2]  589:                    int     msdos
[4MO]  590:                    stc
[CY4]  591:                    jmp     ausgefuehrt
[V10]  592: ;
[D3E]  593: help_screen:       ; ------------------------
       -----------------------------------
[3Z1]  594:                    mov     bh,0
[3G5]  595:                    call    clear_screen
[9ND]  596:                    lea     dx,helpmenue
       ; adresse des helpmenues.
[602]  597:                    mov     ah,9
[BK9]  598:                    int     msdos
       ; display string.
[8D4]  599: help_aus:          stc
[CY4]  600:                    jmp     ausgefuehrt
[V10]  601: ;
[17E]  602: stapeltiefe:       ;------------------------
       -----------------------------------
[9NC]  603:                    mov     dx,offset tiefe1
       ; display message1
[602]  604:                    mov     ah,9
[VY2]  605:                    int     msdos
[4PG]  606:                    mov     al,max_cmdz_puffer
       ; umwandlung in ascii-zahl.
[TM8]  607:                    aam
       ; max '99' moeglich.
[1T2]  608:                    mov     dx,ax
[PX2]  609:                    xchg    dh,dl
[DY1]  610:                    cmp     dl,0
[K76]  611:                    je      fuehrende_null
[BA2]  612:                    add     dl,'0'
[4FA]  613:                    mov     ah,6
       ; al wird zerstoert.
[VY2]  614:                    int     msdos
[T78]  615: fuehrende_null:    mov     dl,dh
[BA2]  616:                    add     dl,'0'
[UZ1]  617:                    mov     ah,6
[P6B]  618:                    int     msdos
       ; display stacktiefe.
[HZC]  619:                    mov     dx,offset tiefe2
       ; display message2.
[602]  620:                    mov     ah,9
[VY2]  621:                    int     msdos
[4MO]  622:                    stc
[CY4]  623:                    jmp     ausgefuehrt
[V10]  624: ;
[75E]  625: disactivate:       ;------------------------
       -----------------------------------
[ND4]  626:                    mov     disable,0h
[602]  627:                    mov     ah,9
[SB8]  628:                    mov     dx,offset disable_m
       sg
[VY2]  629:                    int     msdos
[4MO]  630:                    stc
[CY4]  631:                    jmp     ausgefuehrt
[V10]  632: ;
[YS4]  633: kein_kommando:
[YBC]  634:                    clc
       ; cf=0, melde: irgendwo fehler.
[S15]  635: ausgefuehrt:       ret

[5U4]  636: kommando?          endp
[V10]  637: ;
[PNG]  638: ;=======================================
       ===================================
[GPK]  639: ;          dies ist die umgeleitete interrup
       t-service-routine der
[3Z7]  640: ;          funktion 0Ah von int 21h
[PNG]  641: ;=======================================
       ===================================
[V10]  642: ;
[BOG]  643: newint:
       ; achtung! hier muss mit segment
[81H]  644:                    cmp     cs:byte ptr disable
       ,0dah    ; prefix  (cs:) gearbeitet

[ZAE]  645:                    jne     old_handler
       ; werden, da DS noch falsch.
[UE8]  646:                    cmp     ah,0ah
       ; int 21 fkt 0a ?
[PC2]  647:                    je      new0a
[LN3]  648: old_handler:
[EEG]  649:                    jmp     cs:dword ptr oldint
       ; nein, dann jmp indirekt zur
[NP5]  650:
       ; alten routine.
[YM1]  651: new0a:
[7W8]  652:                    mov     cs:command_ds,ds
       ; ds:dx,
[C4H]  653:                    mov     cs:command_dx,dx
       ; pufferadresse die command.com
[L98]  654:
       ; an fkt 0a uebergab. merken!.
[4Z1]  655:                    push    ax
[5Z1]  656:                    push    bx
[6Z1]  657:                    push    cx
[7Z1]  658:                    push    dx
[SY1]  659:                    push    si
[TO9]  660:                    push    di
       ; rette register!
[YY1]  661:                    push    es
[XY1]  662:                    push    ds
[3XD]  663:  ;                                  !!!!!!!!!!!!!
       !  aktuelles ds und es setzen.
[03D]  664:                    push    cs
       ; muss unbedingt geschehen, da
[1VD]  665:                    pop     ax
       ; das programm nicht vom dos aus
[7ZD]  666:                    mov     ds,ax
       ; gestartet wird, das dann diese
[BXB]  667:                    mov     es,ax
       ; aufgabe uebernimmt. !!!!!!!!
[E9E]  668:                    cld
       ; fuer verschiebungen, vorwaerts!
[W7G]  669:                    mov     cmdz.aktlen,0
       ; puffergroessen initialisieren.
[HF4]  670:                    mov     cmdz_ptr,0
[TBD]  671:                    mov     insflag,0
       ; einfuegemodus loeschen.
[000]  672:
[3Z1]  673:                    mov     bh,0

[BFA]  674:                    mov     ah,3
       ; wo steht der cursor?
[MW2]  675:                    int     video
[KMC]  676:                    mov     curpos,dx
       ; speichere cursorpos.
[000]  677:
[V9F]  678: warte:             mov     ah,1
       ; warte auf tasteneingabe.
[PT1]  679:                    int     16h
[0EC]  680:                    je      warte
       ; zeroflag =1 keine taste.
[2Z1]  681:                    mov     ah,0
[SU7]  682:                    int     16h
       ; hole zeichen
[SWB]  683:                    cmp     al,0
       ; al = 0, dann funkt.-taste.
[ERC]  684:                    je      funkt_taste
       ; al = 0 ==> ah = scan code.
[GP8]  685:                    cmp     al,esc
       ; esc-taste?
[S24]  686:                    je      esc_taste
[SPB]  687:                    cmp     al,backspace
       ; backspace-taste?
[C24]  688:                    je      bs_taste
[B18]  689:                    cmp     al,cr
       ; cr-taste?
[SVC]  690:                    je      cr_taste
       ; zeichen im definierten
[DEB]  691:                    cmp     al,valid_code_min
       ; codebereich?
[6SB]  692:                    jb      notvalid
       ; nein, ignore and peep!
[PY6]  693:                    cmp     al,valid_code_max
[944]  694:                    jna     ablegen
[FC7]  695: notvalid:          jmp     ignore
[V10]  696: ;
[0V2]  697: ablegen:
[VBA]  698:                    cmp     cmdz_ptr,byte ptr c
       mdz_len -1
[R7F]  699:                    jae     ignore
       ; puffer voll? ja, ignore and peep.
[N8B]  700:                    cmp     cmdz.aktlen,byte pt
       r cmdz_len -1
[GF3]  701:                    jae     ignore
[YK6]  702:                    call    zeichenablegen
[CX2]  703:                    jmp     warte
[V10]  704: ;
[D36]  705: ignore:            call    peep
[CX2]  706:                    jmp     warte
```

# DOS EXTRA Nr. 1: Ausverkauft

# DOS EXTRA Nr. 2: Nicht mit Gold aufzuwiegen

## – noch über den Verlag erhältlich –

Best.-Nr. 354

**EXTRA** DOS-Commander: Fensterorientierte Benutzeroberfläche für MS-DOS

**EXTRA** Seitenweise Utilities, die den Umgang mit dem PC erleichtern.

**EXTRA** Mit dem SForth-Compiler 3.1 programmieren Sie in einer neuen Dimension.

**EXTRA** Action, Spaß und Spannung: Superspiele zum Abtippen

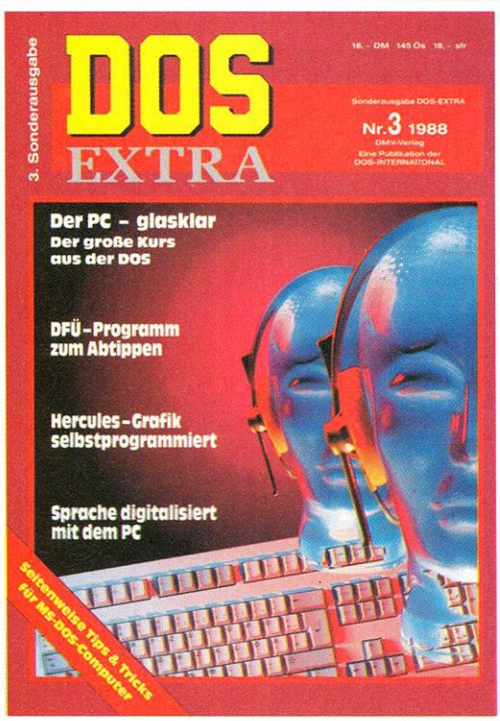
**EXTRA** 3D-CAD-Programm der Superlative.

| Inland: | | Ausland: | |
|---|---|---|---|
| Einzelpreis | 18,– DM | Einzelpreis | 18,– DM |
| zzgl. Versandkosten | 3,– DM | zzgl. Versandkosten | 5,– DM |
| **Endpreis** | **21,– DM** | **Endpreis** | **23,– DM** |

# DOS EXTRA Nr. 3

## Das Heft schlägt beide

## – noch erhältlich –




Best.-Nr. 356

**EXTRA** Der große DOS-Kurs »Der PC – glasklar« in überarbeiteter Form

**EXTRA** Datenfernübertragungs-Programm in 100% Assemblercode

**EXTRA** Zeichensatzeditor für Epson-kompatible Drucker

**EXTRA** Sprachdigitalisierung mit dem PC

**EXTRA** Hercules-Grafik-Toolbox in Turbo Pascal

**EXTRA** Grafikfunktionen für Turbo C

| Inland: | | Ausland: | |
|---|---|---|---|
| Einzelpreis | 18,– DM | Einzelpreis | 18,– DM |
| zzgl. Versandkosten | 3,– DM | zzgl. Versandkosten | 5,– DM |
| **Endpreis** | **21,– DM** | **Endpreis** | **23,– DM** |

Bitte Bestellkarte benutzen

**DMV · Postfach 250 · 3440 Eschwege**

```
[FZM] funkt_taste:    call    funkt_edit
      ; springe zu den funktionstasten-
[G98]                 jmp     warte
      ; routinen.
[OE8] esc_taste:      call    neubeginn
[CX2]                 jmp     warte
[X88] bs_taste:       call    del_left
[CX2]                 jmp     warte
[5P8] cr_taste:       mov     bl,cmdz_ptr
[KN2]                 xor     bh,bh
[DLF]                 mov     cmdz.text[bx],al
      ; Odh in den puffer ablegen.
[5Z1]                 push    bx
[8L1]                 inc     bl
[L7G] ceol:           cmp     cmdz.aktlen,bl
      ; mitten in einer zeile?
[ROE]                 jb      no_ceol
      ; ja, dann clear to end of line.
[6U3]                 call    delkey
[YB2]                 jmp     ceol
[CD5] no_ceol:        pop     bx
[WYE]                 mov     cmdz.aktlen,bl
      ; zeichenanzahl eintragen.
[FAC]                 call    kommando?
      ; kommandos suchen.
[K6F]                 jc      war_kommando
      ; cf gesetzt, wenn kommando.
[UOH]                 call    copy_zeile_comm
      ; kommandozeile in den zeilen-
[6HK]                 call    copy_zeile_stack
      ; puffer von command.com kopieren
[MDE]                 jmp     normal_aus
      ; und in den stack aufnehmen.
[Y74] war_kommando:
[RZ4]                 mov     cmdz.aktlen,0
[O9F]                 mov     cmdz.text,cr
      ; leere eingabe vortaeuschen.
[VE7]                 call    copy_zeile_comm
[5B3] normal_aus:
[8N1]                 pop     ds
[9N1]                 pop     es
[MM1]                 pop     di
[BLD]                 pop     si
      ; registerinhalt restaurieren.
[KN1]                 pop     dx
[HN1]                 pop     cx
[GN1]                 pop     bx
[FN1]                 pop     ax
[C21]                 iret
[V10] ;
[PNG] ;==========================================
      ==================================
[1NA] ;                            puffer und datenb
      ereich
[PNG] ;==========================================
      ==================================
[V10] ;
[7L7] ;' typenvereinbarungen !
[V10] ;
[C3M] zeilenpuffer    struc
      ; so muss der zeilenpuffer fuer die
[9LG]                 maxlen  db    cmdz_len
      ; funktion Oah des interrupts 21h
[XC8]                 aktlen  db    0
      ; aussehen.
[EP3]                 text    db    cr
[PV5]                         db    cmdz_len -1 du
      p (?)
[NG8] zeilenpuffer    ends
      ;
[V10] ;
[X1M] ftastenpuffer   struc
      ; puffertyp fuer die f-tasten und
[67C]                 faktlen  db    0
      ; den kommandostack.
[4R3]                 ftext    db    cr
[PV5]                          db    cmdz_len -1 d
      up (?)
[VZ5] ftastenpuffer   ends
[V10] ;
[6AC] ;------------------------------------------
      ----------------------------------
[V10] ;
[LZE] fcb             db      0,"f_tasteneco"
      ; file control block.
[2E3]                 db      26   dup (?)
[V10] ;
[DHK] cmdz            zeilenpuffer <>
      ; puffer fuer die kommandozeile.
[7OF] cmdz_ptr        db      0
      ; kommandozeilenpointer.
[V10] ;
[O3H] ft10            ftastenpuffer <>
      ; puffer fuer die 10 f-tasten.
[KW8] ft1             ftastenpuffer 9 dup (<>)
[V10] ;
[8AN] cmd_stack       ftastenpuffer (max_cmdz_puf
      fer -1) dup (<>)  ; stack fuer die
[N3L] last_stack_entry ftastenpuffer <>
                       ; kommandozeile.
[6S7] cmd_stack_end    label   word
[4LR] cmd_stack_ptr_e dw       offset cmd_stack
      ; stackpointer fuer eintragungen.
[Z7P] cmd_stack_ptr_a dw       offset cmd_stack
      ; stackpointer fuer ausgaben.
[V10] ;
[72L] oldint           label   byte
      ; hier wird der alte interrupt 21h
[MOK] oldintofs        dw      ?
      ; abgelegt. an seine stelle tritt
[6GH] oldintseg        dw      ?
      ; ein zeiger auf dieses programm.
[V10] ;
[ZVH] curpos           label   word
      ; bilschirm-cursorposition.
[OZ4] curpos_x         db      ?
[OZ4] curpos_y         db      ?
[V10] ;
[XCG] command_ds       dw      ?
      ; save dseg von command.com.
[YVF] command_dx       dw      ?
      ; save dx-reg von command.com.
[V10] ;
[NEG] insflag          db      0
      ; flag fuer den einfuegemodus.
[ONL] disable          db      Odah                 ;
      expacomm nur in funktion, wenn Odah.
[6VK] cmdkeyword       db      "eco"
      ; schluesselwort fuer kommandoer-
[MZM] cmdkeyword_len   dw      $-cmdkeyword
      ; kennung und laenge des wortes.
[V10] ;
[V6F] tiefe1           db      cr,lf,'Die Stapelt
      iefe beträgt ','$'
[WXB] tiefe2           db      ' Eingabezeilen',c
      r,lf,'$'
[V10] ;
[GN6] doppelpunkt      db      ': ','$'
[SW6] init_msg         db      cr,lf
[2VE]                  db      'Expacomm ist inst
      alliert. (c) H. Rieckmann, 1988'
[103]                  db      cr,lf,'$'
[4BN] erneute_inst     db      cr,lf,'Expacomm is
      t bereits installiert !',cr,lf,'$'
[MKK] reaktiviert      db      cr,lf,'Expacomm is
      t reaktiviert !',cr,lf,'$'
[ADM] disable_msg      db      cr,lf,lf,'Expacomm
       ist außer Funktion !',cr,lf,'$'
[1ZG] no_file          db      cr,lf,lf,'File nic
      ht gefunden !',cr,lf,'$'
[V10] ;
[4V6] helpmenue        label   byte
[AA2]  db cr,lf
[2FL]  db '            ╔╤═══════════════════════════
      ═══════════════════════════════╤╗',cr,lf
[4GS]  db '    ║ DOS-Eingabezeilen-Editor ║ HI
      LFE ║   Kommandos von expacomm   ║',cr,lf
[8EL]  db '            ╟┴═══════════════════════════
      ═══════════════════════════════╢',cr,lf
[C1C]  db '    ║
      ║                                  ║',cr,lf
[8SN]  db '    ║    <--    Cursor links
      ║   eco ist der default-Kommando-  ║',cr,lf
[DUM]  db '    ║    -->    Cursor rechts
      ║          name. Kann aber geändert ║',cr,lf
[D5K]  db '    ║    Home   Zeilenanfang
      ║          werden.                  ║',cr,lf
[2BF]  db '    ║    End    Zeilenende
      ║                                  ║',cr,lf
[PCN]  db '    ║    <==    löscht Zeichen links
      ║   eco fx text,  x:0-9, 0=F10     ║',cr,lf
[AOP]  db '    ║    Del    löscht an Cursorpos
      ║          (Progr. der F-Tasten)    ║',cr,lf
[FZF]  db '    ║    Ins    Einfügen ein
      ║                                  ║',cr,lf
[5YK]  db '    ║    Esc    Neubeginn
      ║   eco r   (read F-Tastenfile)    ║',cr,lf
[6CK]  db '    ║    PgUp   Zeigt Tastenbelegung
      ║                                  ║',cr,lf
[N8M]  db '    ║    F1-    vordefiniertes
      ║   eco w   (write F-Tastenfile)   ║',cr,lf
[ABE]  db '    ║    F10    Kommando
      ║                                  ║',cr,lf
[SGG]  db '    ║
      ║   eco t   (zeigt Stapeltiefe)    ║',cr,lf
[K5G]  db '    ║    Cursor hoch / tief
      ║                                  ║',cr,lf
[9AM]  db '    ║           holt ein Kommando
      ║   eco ?   (Dieses Hilfsmenue)    ║',cr,lf
[D7G]  db '    ║           vom Stack, Durchlauf
      ║                                  ║',cr,lf
```

```
[H4M]  829:   db '     ||          in entspr. Richtung
              ||   eco #   (hebt expacomm auf)     ||',cr,lf
[C1C]  830:   db '     ||                                ||',cr,lf
[TFL]  831:   db '     ||                                ||',cr,lf
              ||                                        ||',cr,lf
[2YO]  832:   db '$'
[V1O]  833:   ;
[167]  834:   f_tabelle       label    byte
[YRT]  835:   db '╔═══════════════════════════════════════
              ═══════════════════════════════════════╗',c
              r,lf
[MCC]  836:   db '||                                    ||',c
              r,lf
[98L]  837:   db '||              F u n k t i o n s t a
              s t e n b e l e g u n g                 ||',c
              r,lf
[MCC]  838:   db '||                                    ||',c
              r,lf
[WRT]  839:   db '╚═══════════════════════════════════════
              ═══════════════════════════════════════╝',c
              r,lf
[WP3]  840:   db cr,lf,lf,'$'
[V1O]  841:   ;
[YLG]  842:   last_byte       db      0ffh
              ; letztes byte des residenten
[YM3]  843:
              ; programms.
[V1O]  844:   ;
[6AC]  845:   ;-------------------------------------------
              -----------------------------------
[VSN]  846:   ;        folgender programmteil geht nach d
              em resident machen verloren,
[WTG]  847:   ;        er wird nur fuer die initialisieru
              ng benoetigt.
[6AC]  848:   ;-------------------------------------------
              -----------------------------------
[V1O]  849:   ;
[S7D]  850:   init:           mov     ax,3521h
              ; get intr.-vector 21h.
[YY1]  851:                   push    es
[VY2]  852:                   int     msdos
[3Y6]  853:                   cmp     bx,offset newint
[RX6]  854:                   je      schon_installiert
[UP9]  855:                   mov     oldintofs,bx
              ; save old.
[1A5]  856:                   mov     oldintseg,es
[Z6C]  857:                   mov     dx,offset newint
              ; set new int 21h.
[PW2]  858:                   mov     ax,2521h
[VY2]  859:                   int     msdos
[9N1]  860:                   pop     es
[6O2]  861:                   mov     ah,9
[E67]  862:                   mov     dx,offset init_msg
[VY2]  863:                   int     msdos
[23E]  864:                   mov     ah,31h
              ; terminate but stay resident.
[4H7]  865:                   mov     dx,offset last_byte
[UNC]  866:                   int     27h
              ; wichtig fuer dos-versionen
[DB9]  867:   ;
              ; kleiner als 3.0.
[RY5]  868:   schon_installiert:
[YDH]  869:                   cmp     es:disable,0dah
              ; wenn schon installiert, dann re-
[Y6E]  870:                   jne     aktivieren
              ; installierung vermeiden.
[A2E]  871:                   pop     es
              ; wenn nur vorruebergehend ausser
[8RK]  872:                   mov     dx,offset erneute_i
              nst   ; funktion, dann wieder akti-
[GL7]  873:                   mov     ah,09h
                    ; vieren.
[VY2]  874:                   int     msdos
[DN2]  875:                   xor     ah,ah
[DR8]  876:                   int     msdos
              ; terminate.
[DD3]  877:   aktivieren:
[ERF]  878:                   mov     es:disable,0dah
              ; expacomm wieder aktivieren.
[9N1]  879:                   pop     es
[WH8]  880:                   mov     dx,offset reaktivie
              rt
[FV2]  881:                   mov     ah,09h
[VY2]  882:                   int     msdos
[DN2]  883:                   xor     ah,ah
[DR8]  884:                   int     msdos
              ; terminate.
[V1O]  885:                     ;
[DG3]  886:   cseg            ends
[U64]  887:                   end     expacomm
[92O]  888:   I
```

**Listing. Der Kommando-Editor für MS-DOS**


*** Turbo-XT-Mainboard**
- mit V20/10 MHz
- Phönix-Bios
- 1024 KB RAM (davon 384 KB RAM-Disk)
- nach Norton 4.0!.....DM 999,–

*** Intel-Inboard 386/PC**
- für 8088-Rechner.....DM 1950,–

*** Super-I/O-Karte**

für PC/XT/AT
- 32 I/O-Leitungen
- 3 programmierb. Timer
- 12 Bit A/D und D/A-Wandler
- 8 gemultiplexte analoge Eingänge.....DM 532,–

*** Turbo-Erweiterung**

für PC/XT (nicht Commodore)
- von 4.77 auf 8 MHz
- mit V20/8 und Anleitung...DM 68,–

*** Sprachsynthesizer**
- erzeugt künstliche Sprache mit einfachem Basic-Programm über Druckerschnittstelle.DM 146,–

**Stippler-Elektronik**
Inh. Georg Stippler
Postfach 11 33 – D-8851 Bissingen
Tel. 0 90 05/4 63

**Profi-Tools für QuickBASIC und Turbo BASIC**

Schreiben Sie schnellere, leistungsfähigere und professionellere Programme! Wir helfen Ihnen dabei mit nützlichen Tools.
Zum Beispiel:
- **Toolboxen (Fenstertechnik, Menüs, DOS-Funktionen etc.)**
- **Relationale Datenbank mit komfortablem Masken-Editor**
- **Grafik-Paket (Geschäftsgrafik und Zeichensatz-Generator)**
- **Maus-Unterstützung für Ihre Programme**

Alle Pakete mit ausführlich dokumentierten Quelltexten und Programmbeispielen. Wo erforderlich, kommen schnelle Assembler-Routinen zum Einsatz. Wollen Sie mehr aus Ihrem BASIC-Compiler herausholen? Wir informieren Sie gerne kostenlos!

**Ingenieur-Büro Harald Zoschke**
**Berliner Str.3,**
**D-2306 Schönberg/Holstein**
**Telefon (0 43 44) 61 66**

Eingetr. Warenzeichen: QuickBASIC: Microsoft: Turbo BASIC: Borland

**...make it easy!**

Neueste Software-Versionen von der COMDEX, USA

**PC TOOLS De Luxe**
(Neu: Caching, Text, Data-Recovery, Tuning u.v.m.)
mit dt. Handbuch DM 188,10

**COPY II PC**
(auch für 3½")
Kopierprogramm DM 111,70

**OPTION BOARD De Luxe**
(auch f.3½" u.Apple Mac.) DM 336,30

**1 DIR Plus**
Benutzeroberfläche DM 313,50

**AUTOMENU**
Menugenerator, dt. HB DM 199,50

**BackEZ**
Datensicherung MS-DOS-Format (auch für LAN u. Floppy-Tape)
dt. Handbuch DM 298,00

**PrintQ**
Druck-Spooling auf Platte DM 273,60

**PrintMate**
res. Druckersteuerung (vertikal) DM 210,90

**LaserMate**
res. Druckersteuerung f. Laserdrucker DM 313,50

**Brooklyn Bridge**
Desk-Top-Lap-Top, 5¼"-3½", ser. Kommunikation, einschl. Kabel DM 399,00

**Easy Flow**
Flowchart-Generator DM 427,50

**REFEREE**
Verwaltung res., Software DM 256,90

**Personal VM**
Benutzeroberfläche für VM/CMS DM 883,50

**DESQview 2.0 inkl. QEMM 386**
Multitasking u.v.m. DM 399,00

**The DOCUMENTOR**
(Wallsoft) Analyse u. Dokumentation f. dBase, Clipper, Fox-Base, dt. Handbuch DM 695,40

**SPF/PC 2.0**
Editor MVS/TSO (Neu: Split-Screen, 43/50 Zeilenlänge, Macros wie TSO 2.2) zus. dt. Handbuch DM 758,10

**KEDIT NEU 4.0**
Editor VM/CMS DM 541,50

**Pers. REXX 1.6**
Interpreter u. Sprache (IBM-SAA) DM 541,50

**GML/PC**
(DCF/Script-VS f. Postscript-Laser) auf Anfrage

Händleranfragen erwünscht!

**MCS Werner Kortmann**
**Wilh.-Reuter-Str. 12**
**6239 Eppstein**
**Tel. 0 61 98/3 24 98**

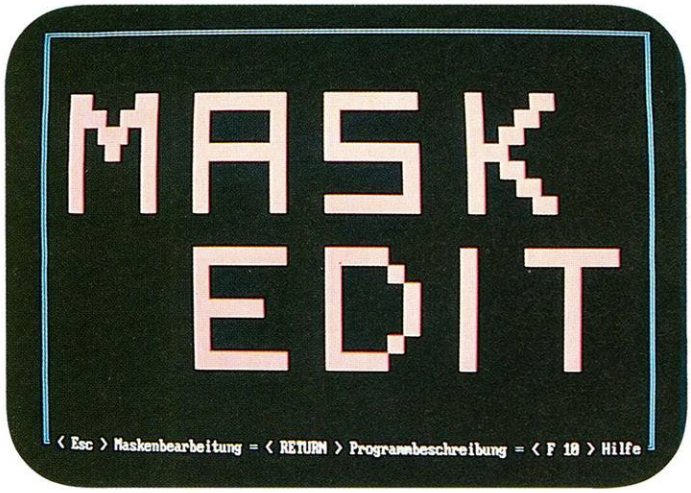



Bild 1. Das Titelbild vom »maskedit«

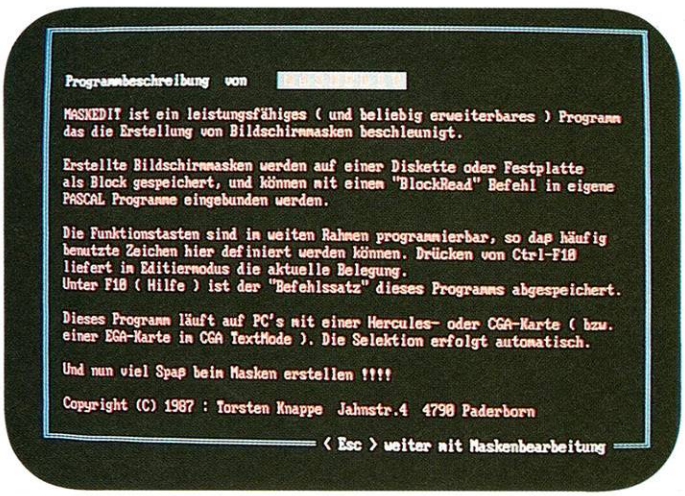



Bild 2. Dieser Hilfsbildschirm (»beschr.scr«) erklärt das Programm

# Hilfsbildschirme – ein Service, der viel bringt

**Eine nutzbringende Einrichtung in ratlosen Momenten sind sogenannte Hilfsbildschirme. Mit diesem Programm können Sie jetzt einfach und schnell solche Masken in Ihre Programme einbauen.**

Das Programm »maskedit« (Listing 1) ist ein leistungsfähiges Utility, das den Programmierer beim Anlegen von Bildschirmmasken unterstützt. Mit »maskedit« erarbeitete Masken sind nicht an eine bestimmte Programmiersprache gebunden. Obwohl das Programm in Turbo Pascal geschrieben ist, lassen sich die Masken beispielsweise auch in C-Programmen einsetzen. Erreicht wird dies durch die Methode, die Maske als Bitmuster zu speichern, und nicht durch spezifische Konstrukte der Programmiersprachen wie etwa die Write-Anweisung von Pascal. Was bringt nun ein Programm wie »maskedit«? Dafür sprechen mehrere Gründe:

Anwendungen, die nicht nur der Programmierer benutzt, müssen über eine leistungsfähige Benutzerschnittstelle verfügen. Die Benutzerschnittstelle ist in der Regel die Bildschirmmaske. Diese sind mehr als nur die Visitenkarte eines Programms. Was nutzen Programmoptionen, die man im Bedarfsfall nicht nutzen kann, weil die Aufrufe nur im Handbuch unter »ferner liefen« beschrieben sind. Des weiteren kann man von einem benutzerfreundlichen Programm erwarten, daß es jederzeit kontextabhängige Hilfstexte anbietet.

Hieraus ergibt sich, daß die Benutzerschnittstelle über den Erfolg eines Programms mitentscheidet. Für den Programmierer bedeutet das eine Menge Arbeit und Zeit, die vor allem in die Bildschirmmaskenerstellung zu investieren ist.

## Fertige Masken werden extern gespeichert und »belasten« aus diesem Grund das Programm nicht

Konventionelle Bildschirmausgaben (in Pascal das erwähnte »write«) sind für einen mit Masken geführten Dialog mit dem Benutzer zu langsam. Wenn der Anwender eines Programms zwischen mehreren Masken blättern muß, sind diese langsamen Bildaufbauzeiten indiskutabel.

Unter Turbo Pascal ist die maximale Codegröße eines Programms auf 64 KByte beschränkt. Da größere Programme doch einiges an Speicher verbrauchen, sollte man es nicht zusätzlich mit Bildschirmmasken »aufblähen«.

»maskedit« genügt den oben angeführten Spezifikationen. Weil »maskedit« sowohl im CGA- als auch im Herculesmodus direkt den Bildschirmspeicher beschreibt (Listing 3), ist die Bildaufbaugeschwindigkeit maximal.

Die fertigen Masken sind auf der Diskette, Festplatte oder RAM-Disk (siehe Listing 2) gespeichert, und daher bleiben im Hauptprogramm nur noch die wesentlichen Routinen. Das Einlesen der Masken erfolgt mit einer kurzen Prozedur, deren Speicherplatzverbrauch nur gering ist. Einzelheiten der Implementierung werden später beschrieben.

## Maskedit ist nicht an eine bestimmte Programmiersprache gebunden, sondern ist universell einsetzbar

»maskedit« ist modular aufgebaut und verständlich dokumentiert, so daß Sie es jederzeit erweitern können. Hier wäre zum Beispiel, neben dem schon implementierten interaktiven Maskendesign, eine interaktive Abfrage der zu benutzenden Datensätze für das Zielprogramm interessant. Nachdem eine Maske spezifiziert ist, könnte eine automatische Datensatzbeschreibung in der Zielsprache (beispielsweise Pascal) erfolgen. Für Applikationen bietet es sich an, die Bildschirmmasken nicht einzeln getrennt auf das Speichermedium zu schreiben, sondern in eine Datei. Ferner könnte man die zweite Bildschirmseite der Grafikkarte nutzen. Der Kreativität des Programmierers sind hier keine Grenzen gesetzt.

Hier noch einige Hinweise für die Implementierung von »maskedit«: Die Prozedur »makemask« ist für die Anferti-

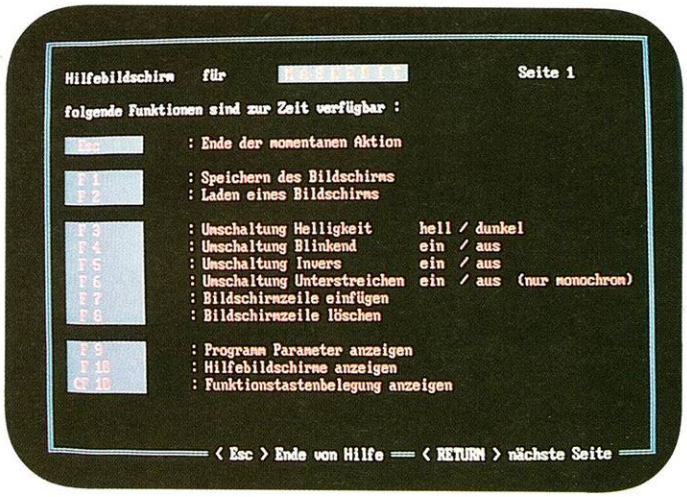



**Bild 3. So sieht die Bildschirmmaske »help1.scr« aus**

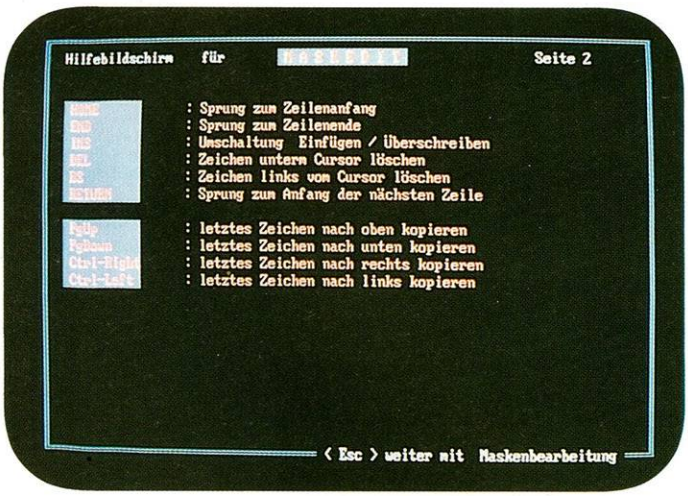



**Bild 4. »help2.scr« zeigt Ihnen den Rest der Tastenbelegung**

gung der Masken zuständig. Der Aufruf erfolgt direkt vom Hauptprogramm, das nur die Initialisierung der globalen Variablen vornimmt und den verwendeten Videomodus feststellt. Der aktuelle Modus wird mit der Funktion »getvideomode« gelesen und zwar durch einen BIOS-Aufruf. In diesem Programm wird entweder Videomodus 3 oder 7 erwartet (Standardtextmodus CGA oder monochrom).

Falls sich der Computer standardmäßig im hochauflösenden Grafikmodus der EGA-Karte (Videomodus 15 oder 16) befindet, muß man »maskedit« so erweitern, daß erst nach Modus 3 oder 7 umgeschaltet wird.

Zuerst initialisiert »makemask« wichtige Bildschirmparameter, die anschließend die Prozedur Parameter anzeigt. Dann folgt eine geschachtelte Repeat-Until-Schleife. Die innere Schleife liest mit der Prozedur »get« eine Tastatureingabe (Listing 4) und führt die damit assoziierte Funktion aus. Ein Betätigen der Taste [F10] (Funktionstaste 10) ruft die Prozedur »helpscreen« auf, die einen Hilfsbildschirm lädt und anzeigt. Das Drücken von »escape« beendet diese Schleife und speichert die gestaltete Maske, wenn es der Benutzer wünscht.

Zwei weitere wichtige Prozeduren sind »writecolor« und »readcolor«, die ein flimmerfreies Schreiben beziehungsweise Lesen des Bildschirmspeichers erlauben. Diese Prozeduren sind als Inline-Anweisungen ausgeführt, da sie sehr zeitkritisch sind. Falls auf den Bildspeicher ohne Beeinflussung der Anzeige zugegriffen werden kann, wird dies durch ein Bit im Statusregister unter der Adresse 3DAhex angezeigt. Die Auswertung dieses Bits ist das Geheimnis eines Bildes ohne Schneegestöber. Im Monochrom-Modus braucht man nicht auf die Synchronisation zu achten. In diesem Modus wird in ein Array geschrieben, das durch eine absolute Adreßangabe den Bildschirmspeicher »überlagert«. Der Bildschirmspeicher ist wortweise (das heißt 16 Bit breit) organisiert, im Highbyte steht das Attribut des Zeichens, im Lowbyte das Zeichen in ASCII-Codierung. Dies erklärt, warum man den Typ Zeichen als varianten Record (oder genauer als »free union«) definiert. Da sowohl »readcolor« als auch »writecolor« wortweise arbeiten, kann man somit mit einem Speicherzugriff Zeichen und Attribut lesen beziehungsweise schreiben. »maskedit« erlaubt im Monochrome-Modus eine freie Kombination der Attribute Intensität, Blinken, Unterstreichen und Invers.

Eine wesentliche Prozedur des Programms »maskedit« ist die Prozedur »getscreen«. Sie liest eine mit »maskedit« erzeugte Bildschirmmaske in eine Variable vom Typ »screen« ein. Im Monochrom-Modus wird diese Variable der Variablen »monoscr« zugewiesen und damit die geladene Maske angezeigt. Im Color-Modus dagegen wird die eingelesene Maske, syntaktisch ein Integer-Array, wortweise mit »writecolor« in den Bildschirmspeicher geschrieben. Wie dies geschieht, zeigt die Prozedur »writesavedpage«.

## Zwei Prozeduren sorgen für eine saubere, flimmerfreie Bildschirmdarstellung

Der Anwender, der mit »maskedit« angelegte Masken nutzen will, muß die Prozedur »getscreen« in sein Programm aufnehmen. Er kann die Einleseprozedur natürlich beliebig erweitern. Die weiteren Prozeduren und Funktionen erledigen im wesentlichen Zulieferaufgaben, ihre Funktionsweise sollte offensichtlich sein.

Wenn der Anwender »maskedit« zum ersten Mal startet, hat er keine fertigen Hilfs- und Erklärungsbildschirme zur Verfügung, so daß sich »maskedit« weder vorstellt noch Hilfstexte anzeigt (siehe Listing 5). Ausführbare Designvorschläge zeigen die Bilder 1 bis 4. Der Anwender kann mit »maskedit« diese oder ähnliche Masken herstellen, »maskedit« erwartet diese Masken unter den Dateinamen »vor«, »beschr«, »help1« und »help2«. Es ist wichtig, »maskedit« die eingegebenen Dateinamen ohne Extension zuzuweisen, da »maskedit« standardmäßig die Endung ».scr« vergibt. Die Bilder 3 und 4 zeigen im Prinzip den Befehlssatz von »maskedit« und wie man diesen aktiviert.

Bild 3, das während des Programmlaufs durch Aktivierung von [Ctrl-F10] auf dem Bildschirm erscheint, zeigt die gültige Belegung der Funktionstasten. Mit der Tastenkombination [Shift-F1] bis [Shift-F6] läßt sich unter anderem ein doppelter Rahmen ziehen. Um nicht jedesmal das Linienzeichen neu eingeben zu müssen, können Sie die Zeichenwiederholfunktion (»Ctrl-Left«, »Ctrl-Right«, »PgUp«, »PgDown«) verwenden. Des weiteren ist jederzeit die direkte Eingabe eines Zeichens als ASCII-Wert möglich. Sie geben dazu bei gedrückter Alt-Taste die Nummer über den Zahlenblock ein.

Mit diesem Programm ergänzen Sie Ihre Programmbibliothek um ein Werkzeug, das Ihnen bei der Entwicklung professioneller Programme hilft, Zeit zu sparen, und es einem »fremden« Anwender erleichtert, mit Ihrem Programm zu arbeiten.

(Torsten Knappe/gh)

```
[FKA]    0: (*****************************************
           ******************************)
[B48]    1: (*                                       *)
[NP9]    2: (*                    M A S K E D I T . P
           A S                                       *)
[B48]    3: (*                                       *)
[F9G]    4: (*  Autor          :   Torsten Knappe Jahns
           tr. 4   4790 Paderborn                    *)
[B48]    5: (*                                       *)
[P1A]    6: (*  Start          :   22.07.87          *)
[AGB]    7: (*  akt. Stand     :   09.09.87          *)
[B48]    8: (*                                       *)
[9HL]    9: (*  Prgm. Beschr.  :  Maskeneditor für Colo
           r Graphics Adapter und                    *)
[DAH]   10: (*                   Monochrome Adapter  (
           Hercules kompatibel )                     *)
[B48]   11: (*                                       *)
[5UD]   12: (* Erstellt in     :   TURBO-PASCAl  V. 3.0
           1 PC-DOS                                  *)
[1DE]   13: (* getestet unter  :   PC-DOS 3.2  und PC-D
           OS 3.3                                    *)
[B48]   14: (*                                       *)
[FKH]   15: (* Rechner         :  AT 03    mit Paradise
           AutoSwitch EGA bzw. Vega EGA *)
[YVB]   16: (*                                   und Colo
           rmonitor                                  *)
[46F]   17: (*                   AT Clone mit Paradise
           AutoSwitch EGA 480 und                    *)
[8YE]   18: (*                                   Color- b
           zw. Monochromemonitor                     *)
[KUH]   19: (*                   XT Clone mit Hercules
           kompatibler Grafikkarte                   *)
[B48]   20: (*                                       *)
[FKA]   21: (*****************************************
           ******************************)
[000]   22:
[RS4]   23: PROGRAM MaskEdit;
[000]   24:
[0C4]   25:   CONST black        =   0;
[EY2]   26:         blue         =   1;
[R93]   27:         green        =   2;
[NY2]   28:         cyan         =   3;
[FM2]   29:         red          =   4;
[674]   30:         magenta      =   5;
[RV3]   31:         yellow       =   6;
[SA3]   32:         white        =   7;
[YX2]   33:         high         =   8;
[VP2]   34:         inv          = 112;
[5B3]   35:         blink        = 128;
[XA8]   36:         MLow         =   2;     { monochrom
           e }
[C63]   37:         MHigh        =  10;
[000]   38:
[KE9]   39:   TYPE  RegType = RECORD CASE Integer OF
[K1F]   40:                     1 : ( ax, bx, cx, dx, b
           p, si, di, ds, es, flags : Integer );
[XXB]   41:                     2 : ( al, ah, bl, bh, c
           l, ch, dl, dh            : Byte    );
[TM0]   42:                   END;
[000]   43:
[G8K]   44:         Zeichen      = RECORD CASE Byte OF
                      { vereinfacht handling}
[BW6]   45:                       1 : ( ch   : Char;
           at  : Byte );
[Y66]   46:                       2 : ( w    : Intege
           r          );
[TM0]   47:                     END;
[000]   48:
[S1H]   49:         ScrTyp       = ARRAY[1..2000] OF Ze
           ichen;            { ein Bildschirm }
[000]   50:
[K05]   51:         Str8         = String[8];
[R55]   52:         Str12        = String[12];
[U75]   53:         Str80        = String[80];
[SP3]   54:
                                { Ord(up) = 0 !! }
[3NH]   55:         Codes        = ( up, down, right, l
           eft, CUp, CDown, CLeft, CRight, cr,
[FMD]   56:                       del, bs, ins, Esc,
           Sonstige, PgUp, PgDown, ToHome,
[V58]   57:                       ToEnd, F1, F2, F3,
           F4, F5, F6, F7, F8, F9, F10,
[PG9]   58:                       SF1, SF2, SF3, SF4
           , SF5, SF6, SF7, SF8, SF9, SF10,
[0X8]   59:                       AF1, AF2, AF3, AF4
           , AF5, AF6, AF7, AF8, AF9, AF10,
[XP9]   60:                       CF1, CF2, CF3, CF4
           , CF5, CF6, CF7, CF8, CF9, CF10   );
```

```
[000]   61:
[P2L]   62:         FktTasten    = ARRAY[Codes] OF Byte
           ;      { Funktionstastenbelegung }
[000]   63:
[000]   64:
[YC6]   65:   VAR   mono             : Boolean;
[HPF]   66:         MonoScr          : ScrTyp ABSOLUTE
           $B000:$0000;          { Monochrom }
[000]   67:
[4P4]   68:         SavedColorScr    ,
[9T6]   69:         SavedMonoScr     : ScrTyp;
[A94]   70:         letzteTaste      ,
[465]   71:         Taste            : Char;
[2T4]   72:         Code             : Codes;
[W08]   73:         FktTastenTab     : FktTasten;
[FR5]   74:         FileName         : Str8;
[5C3]   75:         ok, again        ,
[SB3]   76:         second           ,
[YN3]   77:         HighMod          ,
[D83]   78:         InsMod           ,
[P83]   79:         InvMod           ,
[UR3]   80:         UnderMod         ,
[LR3]   81:         BlinkMod         ,
[NV3]   82:         geaendert        ,
[ZF6]   83:         CursFlag         : Boolean;
[PF7]   84:         AktColor, Px, Py : Integer;
[000]   85:
[SMB]   86:   {-------------------------------------
           ---------------------------------}
[000]   87:
[WGG]   88:   {$Iscreen.inc}         { enthaelt Bildsch
           irmroutinen      }
[RLF]   89:   {$Itast.inc  }         { enthaelt Tastatu
           rroutinen        }
[2RE]   90:   {$Idisk.inc  }         { enthaelt Diskrou
           tinen            }
[R5G]   91:   {$Iuty.inc}            { enthaelt verschi
           edene Prozeduren }
[SMB]   92:   {-------------------------------------
           ---------------------------------}
[000]   93:
[BX4]   94:   PROCEDURE MakeMask;
[000]   95:
[TPH]   96:     { Hauptprozedur, selektiert die gewuens
           chte Aktion }
[000]   97:
[CD5]   98:     VAR tast           : Char;
[000]   99:
[B21]  100:     BEGIN
[NF2]  101:       ClrScr;
[GW8]  102:       BelegFktTasten;                { Init }
[XS3]  103:       InsMod:=True;
[GH4]  104:       HighMod:=True;
[7A4]  105:       InvMod:=False;
[XC5]  106:       UnderMod:=False;
[PC5]  107:       BlinkMod:=False;
[GL3]  108:       again:=False;
[8W3]  109:       second:=True;
[NSF]  110:       Parameter;                    { Anfangsb
           elegung der Fkt-Tasten }
[4L5]  111:       letzteTaste:=Chr(0);
[KF3]  112:       Taste:=Chr(0);
[XG1]  113:       REPEAT
[XG1]  114:         REPEAT
[TM5]  115:           letzteTaste:=Taste;
[GE8]  116:           IF NOT again THEN Get(Taste, Code
           );
[RD2]  117:           CASE CODE OF
[477]  118:             F1             : SaveScreen(Tru
           e);
[L65]  119:             F2             : LoadScreen;
[S08]  120:             F3             : HighMod:=NOT H
           ighMod;
[KY8]  121:             F4             : BlinkMod:=NOT
           BlinkMod;
[CD7]  122:             F5             : InvMod:=NOT In
           vMod;
[25C]  123:             F6             : IF mono THEN U
           nderMod:=NOT UnderMod;
[BK4]  124:             F7             : InsLine;
[CG4]  125:             F8             : DelLine;
[L15]  126:             F9             : Parameter;
[3A5]  127:             F10            : HelpScreen;
[1W7]  128:             CF10           : FktTastenBeleg
           ung;
[000]  129:
[UD5]  130:             CF1            : AktColor:=1;
[1E5]  131:             CF2            : AktColor:=2;
[8E5]  132:             CF3            : AktColor:=3;
[FE5]  133:             CF4            : AktColor:=4;
[PE5]  134:             CF5            : AktColor:=5;
[XE5]  135:             CF6            : AktColor:=6;
[4F5]  136:             CF7            : AktColor:=7;
[N59]  137:             CF8            : AktColor:=0;
           { schwarz }
[G22]  138:             CF9            : ;
[NR5]  139:             Left           : IF WhereX > 1
```

```
[BU9]  140:                                   THEN GotoXY(
       Pred(WhereX), WhereY)
[A97]  141:                                   ELSE GotoXY(
       80, Pred(WhereY));
[CF6]  142:              Right          : IF WhereX < 80
[YU9]  143:                                   THEN GotoXY(
       Succ(WhereX), WhereY)
[BT6]  144:                                   ELSE GotoXY(
       1, Succ(WhereY));
[M75]  145:              Up             : IF WhereY > 1
[SE9]  146:                                   THEN GotoXY(
       WhereX, Pred(WhereY))
[XU5]  147:                                   ELSE GotoXY(
       WhereX, 25);
[F46]  148:              Down           : IF WhereY < 25
[BF9]  149:                                   THEN GotoXY(
       WhereX, Succ(WhereY))
[EW5]  150:                                   ELSE GotoXY(
       WhereX, 1);
[ZH8]  151:              ToHome         : GotoXY(1, Wher
       eY);
[228]  152:              ToEnd          : GotoXY(80, Whe
       reY);
[CR5]  153:              bs             : BackSpace(1);
[KX5]  154:              del            : BackSpace(0);
[ET7]  155:              ins            : InsMod:=NOT In
       sMod;
[RL5]  156:              cr             : IF WhereY < 25
[5G6]  157:                                   THEN GotoXY(
       1, Succ(WhereY))
[3M3]  158:                                   ELSE GotoXY(
       1, 1);
[C24]  159:              SF1..SF10      : BEGIN
[NK9]  160:                                   Taste:=Chr(F
       ktTastenTab[Code]);
[8S4]  161:                                   Code:=Sonsti
       ge;
[U43]  162:                                   again:=True;
[9D4]  163:                                   second:=Fals
       e;
[TM0]  164:                                 END;
[EY3]  165:              AF1..AF10      : BEGIN
[NK9]  166:                                   Taste:=Chr(F
       ktTastenTab[Code]);
[8S4]  167:                                   Code:=Sonsti
       ge;
[U43]  168:                                   again:=True;
[9D4]  169:                                   second:=Fals
       e;
[TM0]  170:                                 END;
[EN4]  171:              CRight         : BEGIN
[Y94]  172:                                   InsMod:=Fals
       e;
[WK5]  173:                                   IF WhereX <
       80 THEN BEGIN
[BA6]  174:                                     GotoXY(Whe
       reX, WhereY);
[HK5]  175:                                     Taste:=let
       zteTaste;
[8S4]  176:                                     Code:=Sons
       tige;
[U43]  177:                                     again:=Tru
       e;
[9D4]  178:                                     second:=Fa
       lse;
[TM0]  179:                                   END;
[TM0]  180:                                 END;
[P64]  181:              CLeft          : BEGIN
[Y94]  182:                                   InsMod:=Fals
       e;
[LT5]  183:                                   IF WhereX >
       1 THEN BEGIN
[S07]  184:                                     GotoXY(Whe
       reX - 2, WhereY);
[HK5]  185:                                     Taste:=let
       zteTaste;
[8S4]  186:                                     Code:=Sons
       tige;
[U43]  187:                                     again:=Tru
       e;
[9D4]  188:                                     second:=Fa
       lse;
[TM0]  189:                                   END;
[TM0]  190:                                 END;
[DV3]  191:              PgUp           : BEGIN
[Y94]  192:                                   InsMod:=Fals
       e;
[X63]  193:                                   IF WhereY >
       1
[49B]  194:                                     THEN GotoX
       Y(Pred(WhereX), Pred(WhereY))
[D58]  195:                                     ELSE GotoX
       Y(Pred(WhereX), 25);
[HK5]  196:                                   Taste:=letzt
       eTaste;
[8S4]  197:                                   Code:=Sonsti
       ge;
```

```
[U43]  198:                                   again:=True;
[9D4]  199:                                   second:=Fals
       e;
[TM0]  200:                                 END;
[XN4]  201:              PgDown         : BEGIN
[000]  202:
[Y94]  203:                                   InsMod:=Fals
       e;
[UG3]  204:                                   IF WhereY <
       25
[KA4]  205:                                     THEN IF Wh
       ereX = 80
[BF9]  206:                                             THE
       N GotoXY(WhereX, Succ(WhereY))
[89B]  207:                                             ELS
       E GotoXY(Pred(WhereX), Succ(WhereY))
[HA4]  208:                                     ELSE IF Wh
       ereX = 80
[KF5]  209:                                             THE
       N GotoXY(WhereX, 1)
[KV7]  210:                                             ELS
       E GotoXY(Pred(WhereX), 1);
[HK5]  211:                                   Taste:=letzt
       eTaste;
[8S4]  212:                                   Code:=Sonsti
       ge;
[U43]  213:                                   again:=True;
[9D4]  214:                                   second:=Fals
       e;
[TM0]  215:                                 END;
[295]  216:              Sonstige       : BEGIN
[AW2]  217:                                   CursorON;
[F63]  218:                                   InsertChar;
[TM0]  219:                                 END;
[TM0]  220:            END;
[597]  221:            IF second THEN again:=False;
[8W3]  222:            second:=True;
[2U3]  223:          UNTIL Code = Esc;
[DR2]  224:          SavePage;
[B13]  225:          CursorOff;
[H03]  226:          GotoXY(1,21);
[R78]  227:          Disp('############################
       ##########################',
[1K4]  228:                black, True, True);
[S1B]  229:          Disp('Programm beenden ?
                           ',
[7R4]  230:                white, False, True);
[FH8]  231:          Disp('############################
       ########### < J / N > ###',
[1K4]  232:                black, True, True);
[B17]  233:          GotoXY(35, Pred(WhereY));
[XG1]  234:          REPEAT
[6Y4]  235:            Get(Taste, Code);
[EN5]  236:            Taste:=UpCase(Taste);
[395]  237:          UNTIL Taste IN ['J', 'N'];
[000]  238:
[EK3]  239:          IF Taste = 'N'
[MH5]  240:            THEN WriteSavedPage
[P07]  241:            ELSE IF geaendert THEN BEGIN
[W83]  242:                   GotoXY(1, 22);
[74H]  243:                   Disp('Sollen die Änderunge
       n gespeichert werden ?             ',
[7R4]  244:                        white, False, True);
[B17]  245:                   GotoXY(35, Pred(WhereY));
[XG1]  246:                   REPEAT
[YL4]  247:                     Get(Tast, Code);
[KZ4]  248:                     Tast:=UpCase(Tast);
[U55]  249:                   UNTIL Tast IN ['J', 'N'];
[WZ5]  250:                   IF tast = 'J' THEN BEGIN
[NF2]  251:                     ClrScr;
[WG5]  252:                     SaveScreen(False);
[E63]  253:                     Taste:=tast;
[TM0]  254:                   END;
[TM0]  255:                 END;
[AW2]  256:          CursorON;
[2K3]  257:        UNTIL Taste = 'J';
[TM0]  258:      END;
[SMB]  259:    {-----------------------------------
       ---------------------------------}
[000]  260:
[H44]  261:    BEGIN      { main }
[NF2]  262:      ClrScr;
[MB3]  263:      AktColor:=3;
[DD5]  264:      geaendert:=False;
[SN4]  265:      CursFlag:=TRUE;
[AW2]  266:      CursorON;
[7B6]  267:      mono:= GetVideoMode = 7;
[YU3]  268:      Vorstellung;
[D6B]  269:      MakeMask;                          { erstell
       e Maske }
[AW2]  270:      CursorON;
[NF2]  271:      ClrScr;
[5L0]  272:    END.
```

**Listing 1. Das Hauptprogramm »maskedit.pas«**

```
[FKA]    0: (*****************************************
           ******************************)
[B48]    1: (*                                       *)
[749]    2: (*                      D I S K . I N C  *)
[B48]    3: (*                                       *)
[F9G]    4: (*  Autor          :   Torsten Knappe Jahns
           tr. 4   4790 Paderborn                     *)
[B48]    5: (*                                       *)
[P1A]    6: (*  Start          :   22.07.87            *)
[3GB]    7: (*  akt. Stand     :   22.09.87            *)
[B48]    8: (*                                       *)
[NNG]    9: (*  Prgm. Beschr.  :  Includedatei fuer MAS
           KEDIT.PAS                                  *)
[B48]   10: (*                                       *)
[FKA]   11: (*****************************************
           ******************************)
[000]   12:
[P8D]   13:   FUNCTION FileExists ( FileName : Str12 )
           : Boolean;
[000]   14:
[7L6]   15:     { Datei vorhanden ? }
[000]   16:
[VN4]   17:     VAR dat : File;
[000]   18:
[B21]   19:     BEGIN
[286]   20:       Assign(dat, FileName);
[PN6]   21:       {$I-} Reset(dat); {$I+}
[R67]   22:       FileExists:=(IOResult = 0);
[GU2]   23:       Close(dat);
[TM0]   24:     END;
[SMB]   25:   {-------------------------------------
           ---------------------------------}
[000]   26:
[ZHB]   27:   PROCEDURE GetScreen ( VAR screen : ScrTyp
           ;
[GZ4]   28:                         VAR ok     : Boolea
           n;
[0G5]   29:                         DateiName  : Str8
           );
[000]   30:
[GTE]   31:     { liest einen Bildschirm von der Disk /
           Platte }
[000]   32:
[WX8]   33:     VAR  ScreenFile    : FILE OF ScrTyp;
[000]   34:
[B21]   35:     BEGIN
[5H9]   36:       ok:=FileExists(DateiName + '.SCR');
[ZU3]   37:       IF ok THEN BEGIN
[VNA]   38:         Assign(ScreenFile, DateiName + '.SC
           R');
[C85]   39:         Reset(ScreenFile);
[KX7]   40:         Read(ScreenFile, screen);
[C75]   41:         Close(ScreenFile);
[TM0]   42:       END;
[TM0]   43:     END;
[SMB]   44:   {-------------------------------------
           ---------------------------------}
[000]   45:
[X28]   46:   PROCEDURE FktTastenBelegung;
[000]   47:
[2HP]   48:     { zeigt die akt. Fkt. Tastenbelegung an
           , und ermöglicht Aenderungen }
[000]   49:
[P48]   50:     VAR l, i, XPos, YPos   : Integer;
[X76]   51:         FTaste,Fsave       : Codes;
[KE6]   52:         at, ofs            : Integer;
[6U9]   53:         FKeyFile           : File OF FktTas
           ten;
[H66]   54:         AltFkt             : Boolean;
[CU4]   55:         rueck              : Byte;
[000]   56:
[R8B]   57:     {-------------------------------------
           ---------------------------------}
[000]   58:
[UCA]   59:     PROCEDURE FktTaste( VAR rueck : Byte );
[000]   60:
[378]   61:       { liest neues Zeichen ein }
[000]   62:
[8V6]   63:       VAR i, j, k, ofs    : Integer;
[1E5]   64:           zw              : String[3];
[000]   65:
[B21]   66:       BEGIN
[MK1]   67:         zw:='';
[493]   68:         GotoXY(1, 24);
[KBE]   69:         Disp('ASCII-Code des Zeichens  --->
           ', white, False, True);
[FL3]   70:         GotoXY(35, 24);
[XG1]   71:         REPEAT
```

```
[FL3]   72:           GotoXY(35, 24);
[B57]   73:           Disp('      ', white, False, True
           );
[MK1]   74:           zw:='';
[FL3]   75:           GotoXY(35, 24);
[XG1]   76:           REPEAT
[6Y4]   77:             Get(Taste, Code);
[RN2]   78:             CASE Code OF
[7M4]   79:               Sonstige  : BEGIN
[SCA]   80:                                 IF (Taste IN ['
           0'..'9']) AND (Length(zw) < 3)
[ZY1]   81:                                   THEN BEGIN
[CS4]   82:                                         zw:=zw
           + Taste;
[N98]   83:                                         ofs:=P
           osition(WhereX, WhereY);
[HN2]   84:                                         IF mon
           o
[DV6]   85:                                           THEN
            WITH MonoScr[ofs] DO
[578]   86:
              BEGIN  ch:=taste; at:=MHigh; END
[B7B]   87:                                           ELSE
            WColor(taste, white + high, ofs);
[1R7]   88:                                         GotoXY
           (Succ(WhereX), WhereY);
[TM0]   89:                                       END;
[TM0]   90:                                 END;
[XS2]   91:               bs          : BEGIN
[3M6]   92:                               IF Length(zw) >
            0 THEN BEGIN
[5R7]   93:                                 GotoXY(Pred(W
           hereX), WhereY);
[N98]   94:                                 ofs:=Position
           (WhereX, WhereY);
[HN2]   95:                                 IF mono
[XP6]   96:                                   THEN MonoSc
           r[ofs].ch:=' '
[L88]   97:                                   ELSE WColor
           (' ', white + high, ofs);
[LN6]   98:                                 Delete(zw, Le
           ngth(zw), 1);
[TM0]   99:                               END;
[TM0]  100:                             END;
[TM0]  101:             END;
[VT8]  102:           UNTIL (Code = cr) AND (Length(zw)
           > 0);
[TK3]  103:           Val(zw, j, k);
[C64]  104:         UNTIL j IN [0..255];
[493]  105:         GotoXY(1, 24);
[GB2]  106:         rueck:=j;
[RBD]  107:         Disp('
                      ', white, False, True);
[TM0]  108:       END;
[R8B]  109:     {-------------------------------------
           ---------------------------------}
[000]  110:
[B21]  111:     BEGIN
[DR2]  112:       SavePage;
[B13]  113:       CursorOff;
[3D3]  114:       FileName:='';
[DZ2]  115:       GotoXY(1, 1);
[ZHA]  116:       Disp('##############################
           ##########################################
           ',
[1K4]  117:            black, True, True);
[UXM]  118:       Disp('Belegung der freiprogrammierbar
           en Funktionstasten                      ',
[7R4]  119:            white, False, True);
[HN2]  120:       IF mono
[31A]  121:         THEN Disp('

           ',
[4S4]  122:                  white, False, False)
[WBL]  123:         ELSE Disp('Die Ctrl F..  Funktionen
           sind nicht veränderbar
           ',
[B15]  124:                  white, False, False);
[ZHA]  125:       Disp('##############################
           ##########################################
           ',
[1K4]  126:            black, True, True);
[HZ1]  127:       IF MONO
[ZY1]  128:         THEN BEGIN
[46D]  129:                Disp('SHIFT F1  :   ' + C
           hr(FktTastenTab[SF1 ]) + '          ' +
[32B]  130:                     'ALT F1  :   ' +  Ch
           r(FktTastenTab[AF1 ]) + '          ' +
[CF7]  131:                     '
           ', white, False, True);
[E6D]  132:                Disp('SHIFT F2  :   ' + C
           hr(FktTastenTab[SF2 ]) + '          ' +
[G2B]  133:                     'ALT F2  :   ' +  Ch
           r(FktTastenTab[AF2 ]) + '          ' +
[CF7]  134:                     '
           ', white, False, True);
```

```
[S6D]  135:                    Disp('SHIFT F3  :   ' + C
       hr(FktTastenTab[SF3 ]) + '          ' +
[X2B]  136:                    'ALT F3  :   ' + Ch
       r(FktTastenTab[AF3 ]) + '          ' +
[CF7]  137:                    '
        ', white, False, True);
[27D]  138:                    Disp('SHIFT F4  :   ' + C
       hr(FktTastenTab[SF4 ]) + '          ' +
[A3B]  139:                    'ALT F4  :   ' + Ch
       r(FktTastenTab[AF4 ]) + '          ' +
[CF7]  140:                    '
        ', white, False, True);
[C7D]  141:                    Disp('SHIFT F5  :   ' + C
       hr(FktTastenTab[SF5 ]) + '          ' +
[R3B]  142:                    'ALT F5  :   ' + Ch
       r(FktTastenTab[AF5 ]) + '          ' +
[CF7]  143:                    '
        ', white, False, True);
[P7D]  144:                    Disp('SHIFT F6  :   ' + C
       hr(FktTastenTab[SF6 ]) + '          ' +
[44B]  145:                    'ALT F6  :   ' + Ch
       r(FktTastenTab[AF6 ]) + '          ' +
[CF7]  146:                    '
        ', white, False, True);
[08D]  147:                    Disp('SHIFT F7  :   ' + C
       hr(FktTastenTab[SF7 ]) + '          ' +
[H4B]  148:                    'ALT F7  :   ' + Ch
       r(FktTastenTab[AF7 ]) + '          ' +
[CF7]  149:                    '
        ', white, False, True);
[A8D]  150:                    Disp('SHIFT F8  :   ' + C
       hr(FktTastenTab[SF8 ]) + '          ' +
[Y4B]  151:                    'ALT F8  :   ' + Ch
       r(FktTastenTab[AF8 ]) + '          ' +
[CF7]  152:                    '
        ', white, False, True);
[M8D]  153:                    Disp('SHIFT F9  :   ' + C
       hr(FktTastenTab[SF9 ]) + '          ' +
[B5B]  154:                    'ALT F9  :   ' + Ch
       r(FktTastenTab[AF9 ]) + '          ' +
[CF7]  155:                    '
        ', white, False, True);
[4CD]  156:                    Disp('SHIFT F10 :   ' + C
       hr(FktTastenTab[SF10]) + '          ' +
[L5B]  157:                    'ALT F10 :   ' + Ch
       r(FktTastenTab[AF10]) + '          ' +
[CF7]  158:                    '
        ', white, False, True);
[DD0]  159:                  END
[YY1]  160:              ELSE BEGIN
[46D]  161:                    Disp('SHIFT F1  :   ' + C
       hr(FktTastenTab[SF1 ]) + '          ' +
[32B]  162:                    'ALT F1  :   ' + Ch
       r(FktTastenTab[AF1 ]) + '          ' +
[399]  163:                    'Ctrl F1  :   Blau
        ', white, False, True);
[E6D]  164:                    Disp('SHIFT F2  :   ' + C
       hr(FktTastenTab[SF2 ]) + '          ' +
[G2B]  165:                    'ALT F2  :   ' + Ch
       r(FktTastenTab[AF2 ]) + '          ' +
[SA9]  166:                    'Ctrl F2  :   Grün
        ', white, False, True);
[S6D]  167:                    Disp('SHIFT F3  :   ' + C
       hr(FktTastenTab[SF3 ]) + '          ' +
[X2B]  168:                    'ALT F3  :   ' + Ch
       r(FktTastenTab[AF3 ]) + '          ' +
[5Y9]  169:                    'Ctrl F3  :   Türkis
        ', white, False, True);
[27D]  170:                    Disp('SHIFT F4  :   ' + C
       hr(FktTastenTab[SF4 ]) + '          ' +
[A3B]  171:                    'ALT F4  :   ' + Ch
       r(FktTastenTab[AF4 ]) + '          ' +
[W79]  172:                    'Ctrl F4  :   Rot
        ', white, False, True);
[C7D]  173:                    Disp('SHIFT F5  :   ' + C
       hr(FktTastenTab[SF5 ]) + '          ' +
[R3B]  174:                    'ALT F5  :   ' + Ch
       r(FktTastenTab[AF5 ]) + '          ' +
[N6A]  175:                    'Ctrl F5  :   Magent
       a', white, False, True);
[P7D]  176:                    Disp('SHIFT F6  :   ' + C
       hr(FktTastenTab[SF6 ]) + '          ' +
[44B]  177:                    'ALT F6  :   ' + Ch
       r(FktTastenTab[AF6 ]) + '          ' +
[B99]  178:                    'Ctrl F6  :   Gelb
        ', white, False, True);
[08D]  179:                    Disp('SHIFT F7  :   ' + C
       hr(FktTastenTab[SF7 ]) + '          ' +
[H4B]  180:                    'ALT F7  :   ' + Ch
       r(FktTastenTab[AF7 ]) + '          ' +
[PM9]  181:                    'Ctrl F7  :   Weiß
        ', white, False, True);
[A8D]  182:                    Disp('SHIFT F8  :   ' + C
       hr(FktTastenTab[SF8 ]) + '          ' +
[Y4B]  183:                    'ALT F8  :   ' + Ch
       r(FktTastenTab[AF8 ]) + '          ' +
```

```
[KBA]  184:                    'Ctrl F8  :   Schwar
       z', white, False, True);
[M8D]  185:                    Disp('SHIFT F9  :   ' + C
       hr(FktTastenTab[SF9 ]) + '          ' +
[B5B]  186:                    'ALT F9  :   ' + Ch
       r(FktTastenTab[AF9 ]) + '          ' +
[CF7]  187:                    '
        ', white, False, True);
[4CD]  188:                    Disp('SHIFT F10 :   ' + C
       hr(FktTastenTab[SF10]) + '          ' +
[L5B]  189:                    'ALT F10 :   ' + Ch
       r(FktTastenTab[AF10]) + '          ' +
[CF7]  190:                    '
        ', white, False, True);
[TM0]  191:                  END;
[000]  192:
[ZHA]  193:       Disp('################################
       ########################################
       ',
[1K4]  194:            black, True, True);
[U7C]  195:       Disp('Sie können mit :', white, False
       , True);
[6H7]  196:       GotoXY(WhereX, Succ(WhereY));
[AXK]  197:       Disp('< SF1 > .. < AF10 > : die Beleg
       ung der Funktionstasten ändern',
[7R4]  198:            white, False, True);
[ZXP]  199:       Disp('< L >                : eine bere
       its erstellte Funktionstastenbelegung laden'
       ,
[7R4]  200:            white, False, True);
[4BN]  201:       Disp('< S >                : die momen
       tane Funktionstastenbelegung speichern',
[7R4]  202:            white, False, True);
[9ZC]  203:       Disp('< Esc >              : zum Edito
       r zurückkehren',
[7R4]  204:            white, False, True);
[XTH]  205:       Disp('#### weiter mit < Esc > oder <
       SF1 > .. < AF10 > oder < L > oder < S > ###'
       ,
[1K4]  206:            black, True, True);
[000]  207:
[XG1]  208:       REPEAT
[XG1]  209:         REPEAT
[6Y4]  210:           Get(Taste, Code);
[M18]  211:         UNTIL ( Code IN [ SF1 .. AF10, Esc
       ]) OR
[LSE]  212:                ( (Code = Sonstige) AND (UpC
       ase(taste) IN  [ 'L', 'S']) );
[RY2]  213:         Case Code OF
[3A3]  214:           SF1..AF10   : BEGIN
[H18]  215:                           Fsave:=Code;    {
        muss sein ! }
[HX8]  216:                           AltFkt := Ord(Cod
       e) > 37;  { AF1 - AF10 }
[BB2]  217:                           IF AltFkt
[4N7]  218:                             THEN YPos:=Ord(
       Code) - 39    { AF1 }
[TP7]  219:                             ELSE YPos:=Ord(
       Code) - 29;   { SF1 }
[5Y8]  220:                           IF AltFkt THEN XP
       os:=38 ELSE XPos:=15;
[AW2]  221:                           CursorON;
[U85]  222:                           FktTaste(rueck);
[NG8]  223:                           FktTastenTab[Fsav
       e]:=rueck;
[B03]  224:                           CursorOFF;
[S18]  225:                           ofs:=Position(XPo
       s, YPos + 6);
[HN2]  226:                           IF mono
[NSD]  227:                             THEN MonoScr[of
       s].ch:=Chr(FktTastenTab[Fsave])
[VLG]  228:                             ELSE WColor(Chr
       (FktTastenTab[Fsave]),high + white, ofs);
[TM0]  229:                         END;
[2K9]  230:           Sonstige    : CASE UpCase(Taste)
       OF
[7T1]  231:                           'L' : BEGIN
[493]  232:                                   GotoXY(1,
        24);
[36F]  233:                                   Disp('Dat
       einame der Definition', white, False, True);
[AL3]  234:                                   GotoXY(30
       , 24);
[AW2]  235:                                   CursorON;
[LH3]  236:                                   GetFileNa
       me;
[B03]  237:                                   CursorOFF
       ;
[MTA]  238:                                   IF FileEx
       ists(FileName + '.SFK') THEN BEGIN
[GM9]  239:                                     Assign(
       FKeyFile, FileName + '.SFK');
[DM4]  240:                                     Reset(F
       KeyFile);
[DX7]  241:                                     Read(FK
       eyFile, FktTastenTab);
```

```
[DL4]  242:                                        Close(F
       KeyFile);
[000]  243:
[ZV0]  244:                                        i:=1;
[FC7]  245:                                        FOR FTa
       ste:=SF1 TO SF10 DO BEGIN
[CU5]  246:                                          ofs:=
       Position(15,i + 4);
[HN2]  247:                                          IF mo
       no
[5D6]  248:                                            THE
       N MonoScr[ofs].ch:=
[V28]  249:
         Chr(FktTastenTab[FTaste])
[K68]  250:                                            ELS
       E WColor(Chr(FktTastenTab
[SB9]  251:
            [FTaste]), white + high, ofs);
[WW2]  252:                                          i:=Su
       cc(i);
[TM0]  253:                                        END;
[ZV0]  254:                                        i:=1;
[D77]  255:                                        FOR FTa
       ste:=AF1 TO AF10 DO BEGIN
[FZ5]  256:                                          ofs:=
       Position(38,i+ 4);
[HN2]  257:                                          IF mo
       no
[5D6]  258:                                            THE
       N MonoScr[ofs].ch:=
[V28]  259:
         Chr(FktTastenTab[FTaste])
[K68]  260:                                            ELS
       E WColor(Chr(FktTastenTab
[SB9]  261:
            [FTaste]), white + high, ofs);
[WW2]  262:                                          i:=Su
       cc(i);
[TM0]  263:                                          END;
[DD0]  264:                                        END
[R74]  265:                                        ELSE Code
       :=Esc;
[TM0]  266:                                    END;
[NT1]  267:                                'S' : BEGIN
[493]  268:                                        GotoXY(1,
        24);
[MN9]  269:                                        Disp('Dat
       einame der Definition',
[7R4]  270:                                             whit
       e, False, True);
[AL3]  271:                                        GotoXY(30
       , 24);
[AW2]  272:                                        CursorON;
[LH3]  273:                                        GetFileNa
       me;
[B03]  274:                                        CursorOFF
       ;
[478]  275:                                        IF Length
       (FileName) > 0 THEN BEGIN
[GM9]  276:                                          Assign(
       FKeyFile, FileName + '.SFK');
[7T5]  277:                                          Rewrite
       (FKeyFile);
[F09]  278:                                          Write(F
       KeyFile, FktTastenTab);
[DL4]  279:                                          Close(F
       KeyFile);
[TM0]  280:                                        END;
[TM0]  281:                                    END;
[TM0]  282:                              END;
[TM0]  283:         END;
[493]  284:       GotoXY(1, 24);
[S7C]  285:       Disp('
                      ', white, False, True);
[2U3]  286:     UNTIL Code = Esc;
[R35]  287:     WriteSavedPage;
[AW2]  288:     CursorON;
[8C2]  289:     Code:=CF10;
[TM0]  290:   END;
[SMB]  291:   {-----------------------------------
       ---------------------------------}
[000]  292:
[RWB]  293:   PROCEDURE SaveScreen( sichern : Boolean )
       ;
[000]  294:
[2LD]  295:     { schreiben des akt. Bildschirmes auf d
       isk }
[000]  296:
[BU9]  297:     VAR  ScreenFile              : FILE OF S
       crTyp;
[PT5]  298:          i, j                    : Integer;
[GGA]  299:          ueberschreiben, fertig : Boolean;
[000]  300:
[B21]  301:     BEGIN
[BA7]  302:       IF sichern THEN SavePage;
```

```
[B03]  303:       CursorOFF;
[PE4]  304:       fertig:=False;
[MC6]  305:       WHILE NOT fertig DO BEGIN
[DZ2]  306:       GotoXY(1, 1);
[NF9]  307:         Disp('##############################
       ##################################',
[1K4]  308:                black, True, True);
[S8F]  309:         Disp('Speichern des Bildschirminhal
       tes                              ',
[7R4]  310:              white, False, True);
[NF9]  311:         Disp('##############################
       ##################################',
[1K4]  312:                black, True, True);
[9Y9]  313:         Disp('Dateiname :
                                        ',
[7R4]  314:              white, False, True);
[ERB]  315:         Disp('##############################
       ###############< Esc > = Abbruch ###',
[1K4]  316:                black, True, True);
[F83]  317:         GotoXY(15, 4);
[AW2]  318:         CursorON;
[LH3]  319:         GetFileName;
[B03]  320:         CursorOFF;
[1R9]  321:         IF Code = Esc THEN fertig:=True;
[746]  322:         IF Code <> Esc THEN BEGIN
[ES6]  323:           ueberschreiben:=True;
[FXA]  324:           IF FileExists(Filename + '.SCR')
       THEN BEGIN
[ZZ2]  325:             GotoXY(1, 7);
[NF9]  326:             Disp('##############################
       ########################################',
[1K4]  327:                    black, True, True);
[0U8]  328:             Disp('
                                                ',
[1K4]  329:                  black, True, True);
[W9L]  330:             Disp('A C H T U N G !!!!  Datei
        existiert bereits :  Überschreiben  ?   ',
[CM4]  331:                  white, True, True);
[0U8]  332:             Disp('
                                                ',
[1K4]  333:                  black, True, True);
[XX9]  334:             Disp('##############################
       ########################## < J / N > ###',
[1K4]  335:                 black, True, True);
[XG1]  336:             REPEAT
[6Y4]  337:               Get(Taste, Code);
[EN5]  338:               Taste:=UpCase(Taste);
[395]  339:             UNTIL Taste IN ['J', 'N'];
[4K8]  340:             ueberschreiben:= taste = 'J';
[2A5]  341:             fertig:=taste = 'J';
[TM0]  342:           END;
[NE5]  343:           IF ueberschreiben
[ZY1]  344:             THEN BEGIN
[Z1A]  345:                    Assign(ScreenFile, FileN
       ame + '.SCR');
[876]  346:                    Rewrite(ScreenFile);
[HN2]  347:                    IF mono
[N8A]  348:                      THEN Write(ScreenFile,
        SavedMonoScr)
[TZA]  349:                      ELSE Write(ScreenFile,
        SavedColorScr);
[C75]  350:                    Close(ScreenFile);
[NX3]  351:                    fertig:=True;
[DD0]  352:                  END
[YY1]  353:             ELSE BEGIN
[ZZ2]  354:                    GotoXY(1, 7);
[DA3]  355:                    FOR j:=1 TO 5 DO
[0U8]  356:                      Disp('


[WM4]  357:                           black, False, Tru
       e);
[TM0]  358:                  END;
[TM0]  359:         END;
[TM0]  360:       END;
[NP9]  361:       IF sichern THEN WriteSavedPage;
[AW2]  362:       CursorON;
[P42]  363:       Code:=F1;
[TM0]  364:     END;
[SMB]  365:   {-----------------------------------
       ---------------------------------}
```

**Listing 2. Die Include-Datei »disk.inc« übernimmt das Speichern und Laden auf Diskette**

```
[000]    0:
[FKA]    1: (*****************************************
           ******************************)
[B48]    2: (*                                        *)
[729]    3: (*                        S C R E E N . I N C
                                                     *)
[B48]    4: (*                                        *)
[F9G]    5: (*  Autor           :   Torsten Knappe Jahns
           tr. 4   4790 Paderborn                    *)
[B48]    6: (*                                        *)
[P1A]    7: (*  Start           :   22.07.87
                                                     *)
[AGB]    8: (*  akt. Stand      :   09.09.87
                                                     *)
[B48]    9: (*                                        *)
[E8M]   10: (*  Prgm. Beschr.   :  Includedatei mit Bild
           schirmroutinen fuer                       *)
[9D9]   11: (*                        MASKED.PAS.
                                                     *)
[B48]   12: (*                                        *)
[FKA]   13: (*****************************************
           ******************************)
[000]   14:
[000]   15:
[000]   16:
[ZBC]   17: PROCEDURE WriteColor ( ca , Offset : Intege
           r );
[000]   18:
[L5L]   19:     { schreibt ohne Schnee direkt in den Bi
           ldwiederholspeicher }
[5RD]   20:     { ca enthaelt das Zeichen + Attribut
                                                     }
[000]   21:
[B21]   22:     BEGIN
[HR1]   23:       INLINE (
[399]   24:                    $8B / $4E / $04 /
                {       MOV CX   , [BP+4]       }
[PZ7]   25:                    $D1 / $E1 /
                {       SHL CX   , 1            }
[XU7]   26:                    $49 /
                {       DEC CX                  }
[XU7]   27:                    $49 /
                {       DEC CX                  }
[T58]   28:                    $8B / $F9 /
                {       MOV DI   , CX           }
[T99]   29:                    $8B / $5E / $06 /
                {       MOV BX   , [BP+6]       }
[U29]   30:                    $B8 / $00 / $B8 /
                {       MOV AX   , B800h        }
[B78]   31:                    $8E / $C0 /
                {       MOV ES   , AX           }
[NX8]   32:                    $BA / $DA / $03 /
                {       MOV DX   , 3DAh         }
[PA8]   33:                    $EC /
                { L1:   IN  AL   , DX           }
[XX7]   34:                    $D0 / $D8 /
                {       RCR AL   , 1            }
[KP7]   35:                    $72 / $FB /
                {       JC  L1                  }
[8G7]   36:                    $FA /
                {       CLI                     }
[UA8]   37:                    $EC /
                { L2:   IN  AL   , DX           }
[X28]   38:                    $D0 / $D8 /
                {       RCR Al   , 1            }
[TU7]   39:                    $73 / $FB /
                {       JNZ L2                  }
[8D7]   40:                    $26 /
                {       ES:                     }
[BW8]   41:                    $89 / $1D /
                {       MOV [DI], BX            }
[7G7]   42:                    $FB                      );
                {       STI                     }
[TM0]   43:     END;
[SMB]   44:   {-------------------------------------
           ---------------------------------}
[000]   45:
[84D]   46:   FUNCTION ReadColor ( Offset : Integer ) :
           Integer;
[000]   47:
[AZF]   48:     { liest direkt aus dem  Bildwiederholsp
           eicher }
[NGD]   49:     { geliefert wird Zeichen + Attribut
                }
[000]   50:
[B21]   51:     BEGIN
[PCB]   52:       INLINE ( $8B / $4E / $04 /
                {       MOV CX   , [BP+4]       }
[PZ7]   53:                    $D1 / $E1 /
                {       SHL CX   , 1            }
[XU7]   54:                    $49 /
                {       DEC CX                  }
[XU7]   55:                    $49 /
                {       DEC CX                  }
[T58]   56:                    $8B / $F9 /
                {       MOV DI   , CX           }
[U29]   57:                    $B8 / $00 / $B8 /
                {       MOV AX   , B800h        }
[B78]   58:                    $8E / $C0 /
                {       MOV ES   , AX           }
[NX8]   59:                    $BA / $DA / $03 /
                {       MOV DX   , 3DAh         }
[PA8]   60:                    $EC /
                { L1:   IN  AL   , DX           }
[XX7]   61:                    $D0 / $D8 /
                {       RCR AL   , 1            }
[KP7]   62:                    $72 / $FB /
                {       JC  L1                  }
[8G7]   63:                    $FA /
                {       CLI                     }
[UA8]   64:                    $EC /
                { L2:   IN  AL   , DX           }
[XX7]   65:                    $D0 / $D8 /
                {       RCR AL   , 1            }
[TU7]   66:                    $73 / $FB /
                {       JNZ L2                  }
[8D7]   67:                    $26 /
                {       ES:                     }
[XN8]   68:                    $8B / $05 /
                {       MOV AX   , [DI]         }
[LG7]   69:                    $FB /
                {       STI                     }
[6C9]   70:                    $89 / $46 / $06      );
                {       MOV [BP+6] , AX         }
[TM0]   71:     END;
[SMB]   72:   {-------------------------------------
           ---------------------------------}
[000]   73:
[7WG]   74:   PROCEDURE WColor ( ch : Char; at : Byte;
           Offset : Integer );
[000]   75:
[D1M]   76:     { formt das Zeichen und Attribut so um,
           dass WriteColor zum }
[3BE]   77:     { schreiben benutzt werden kann
                                                     }
[000]   78:
[GM4]   79:     VAR ca : Integer;
[000]   80:
[B21]   81:     BEGIN
[1V5]   82:       ca:=Ord(ch) + at Shl 8;
[MR6]   83:       WriteColor(ca, Offset);
[TM0]   84:     END;
[SMB]   85:   {-------------------------------------
           ---------------------------------}
[000]   86:
[P38]   87:   FUNCTION GetVideoMode : Byte;
[000]   88:
[SYC]   89:     { DOS - Aufruft zum Bildschirmmode lese
           n }
[000]   90:
[025]   91:     VAR Reg : RegType;
[000]   92:
[B21]   93:     BEGIN
[0K2]   94:       Reg.ah:=15;
[CP3]   95:       Intr($10, Reg);
[336]   96:       GetVideoMode:=Reg.al;
[TM0]   97:     END;
[SMB]   98:   {-------------------------------------
           ---------------------------------}
[000]   99:
[1F5]  100:   PROCEDURE CursorOFF;
[000]  101:
[025]  102:     VAR Reg : RegType;
[000]  103:
[B21]  104:     BEGIN
[526]  105:       IF CursFlag THEN BEGIN
[VC2]  106:         Reg.ah:=1;
[NZ9]  107:         Reg.ch:=$20;                    { Cu
           rsor aus   }
[CP3]  108:         Intr($10, Reg);
[UUH]  109:         CursFlag:=FALSE;                { Cu
           rsor im Moment nicht sichtbar }
[TM0]  110:       END;
[TM0]  111:     END;
[SMB]  112:   {-------------------------------------
           ---------------------------------}
[000]  113:
[475]  114:   PROCEDURE CursorON;
[000]  115:
[025]  116:     VAR Reg : RegType;
[000]  117:
[B21]  118:     BEGIN
[CD6]  119:       IF NOT CursFlag THEN BEGIN
[VC2]  120:         Reg.ah:=1;
[HN2]  121:         IF mono
[ZY1]  122:           THEN BEGIN
[A9E]  123:                  Reg.cl:=12;            { se
```

```
          tze normale Cursorgroesse MGA }
[6K2]  124:                   Reg.ch:=13;
[DD0]  125:                 END
[YY1]  126:               ELSE BEGIN
[LHE]  127:                   Reg.cl:=7;              { se
          tze normale Cursorgroesse CGA }
[CD2]  128:                   Reg.ch:=8;
[TM0]  129:                 END;
[CP3]  130:           Intr($10, Reg);
[VEC]  131:           CursFlag:=TRUE;                 { Cu
          rsor sichtbar }
[TM0]  132:         END;
[TM0]  133:       END;
[SMB]  134:    {-----------------------------------------
          --------------------------------}
[000]  135:
[HZ4]  136:    PROCEDURE SavePage;
[000]  137:
[LWD]  138:      { sichert die angezeigte Bildschirmseite
          }
[000]  139:
[5K4]  140:      VAR i : Integer;
[000]  141:
[B21]  142:      BEGIN
[HN2]  143:        IF mono
[TP7]  144:          THEN SavedMonoScr:=MonoScr
[KWE]  145:          ELSE FOR i:=1 TO 2000 DO SavedColor
          Scr[i].w:=ReadColor(i);
[EW2]  146:        Px:=WhereX;
[KW2]  147:        Py:=WhereY;
[NF2]  148:        ClrScr;
[TM0]  149:      END;
[SMB]  150:    {-----------------------------------------
          --------------------------------}
[000]  151:
[TF7]  152:    PROCEDURE WriteSavedPage;
[000]  153:
[FHD]  154:      { schreibt die gesicherte Seite zurueck
          }
[000]  155:
[5K4]  156:      VAR i : Integer;
[000]  157:
[B21]  158:      BEGIN
[HN2]  159:        IF mono
[CF7]  160:          THEN MonoScr:=SavedMonoScr
[M8F]  161:          ELSE FOR i:=1 TO 2000 DO WriteColor
          (SavedColorScr[i].w, i);
[344]  162:        GotoXY(Px, Py);
[TM0]  163:      END;
[SMB]  164:    {-----------------------------------------
          --------------------------------}
[000]  165:
[SWC]  166:    Function GetCharAt(ofs : Integer): Byte;
[000]  167:
[WFG]  168:      { liefert das Attribut des an Ofs stehe
          nden Zeichens }
[000]  169:
[B21]  170:      BEGIN
[5R9]  171:        GetCharAt:=ReadColor(ofs) AND $FF00;
[TM0]  172:      END;
[SMB]  173:    {-----------------------------------------
          --------------------------------}
[000]  174:
[HSC]  175:    FUNCTION Position ( x, y : Integer ) : In
          teger;
[000]  176:
[GS9]  177:      BEGIN Position:=(Pred(y) * 80) + x;  EN
          D;
[000]  178:
[SMB]  179:    {-----------------------------------------
          --------------------------------}
[000]  180:
[Y77]  181:    FUNCTION Attribut : Byte;
[000]  182:
[9MK]  183:      { bei inverser Darstellung wirkt Intens
          itaetsbit nicht }
[V8F]  184:      { bei unterstreichen wirkt invers nicht
                  }
[000]  185:
[B21]  186:      BEGIN
[HN2]  187:        IF mono
[HG4]  188:          THEN IF highMod
[9W3]  189:                 THEN IF invMod
[6K4]  190:                        THEN IF blinkMod
[DH4]  191:                               THEN IF UnderM
          od
[0F6]  192:                                      THEN At
          tribut:= 153 { 141 }
[ME6]  193:                                      ELSE At
          tribut:= 240 { 140 }
[CH4]  194:                               ELSE IF UnderM
          od
[ZF4]  195:                                      THEN At
          tribut:= 9
[ER4]  196:                                      ELSE At
          tribut:= 112
[5K4]  197:                               ELSE IF blinkMod
[DH4]  198:                                      THEN IF UnderM
          od
[UR4]  199:                                      THEN At
          tribut:= 153
[TE6]  200:                                      ELSE At
          tribut:= 140 { 136 }
[CH4]  201:                               ELSE IF UnderM
          od
[HP4]  202:                                      THEN At
          tribut:=   9
[RP4]  203:                                      ELSE At
          tribut:=  10
[8W3]  204:                 ELSE IF invMod
[6K4]  205:                        THEN IF blinkMod
[DH4]  206:                               THEN IF UnderM
          od
[YR4]  207:                                      THEN At
          tribut:= 145
[FR4]  208:                                      ELSE At
          tribut:= 240
[CH4]  209:                               ELSE IF UnderM
          od
[RF4]  210:                                      THEN At
          tribut:= 1
[ER4]  211:                                      ELSE At
          tribut:= 112
[5K4]  212:                               ELSE IF blinkMod
[DH4]  213:                                      THEN IF UnderM
          od
[YR4]  214:                                      THEN At
          tribut:= 145
[4P6]  215:                                      ELSE At
          tribut:= 130  { 128 }
[CH4]  216:                               ELSE IF UnderM
          od
[RF4]  217:                                      THEN At
          tribut:= 1
[SS8]  218:                                      ELSE At
          tribut:= 7          { normal }
[GG4]  219:          ELSE IF highMod
[9W3]  220:                 THEN IF invMod
[6K4]  221:                        THEN IF blinkMod
[35D]  222:                               THEN Attribut:
          = AktColor + high + inv + blink
[H3B]  223:                               ELSE Attribut:
          = AktColor + high + inv
[5K4]  224:                        ELSE IF blinkMod
[NCC]  225:                               THEN Attribut:
          = AktColor + high + blink
[5P9]  226:                               ELSE Attribut:
          = AktColor + high
[8W3]  227:                   ELSE IF invMod
[6K4]  228:                          THEN IF blinkMod
[TDB]  229:                                 THEN Attribut:
          = AktColor + inv + blink
[439]  230:                                 ELSE Attribut:
          = AktColor + inv
[5K4]  231:                          ELSE IF blinkMod
[LR9]  232:                                 THEN Attribut:
          = AktColor + blink
[PC7]  233:                                 ELSE Attribut:
          = AktColor ;
[TM0]  234:      END;
[SMB]  235:    {-----------------------------------------
          --------------------------------}
[000]  236:
[Z0A]  237:    PROCEDURE BackSpace ( rueck : Integer );
[000]  238:
[XY5]  239:      VAR i, Px, Py  : Integer;
[000]  240:
[B21]  241:      BEGIN
[EW2]  242:        Px:=WhereX;
[KW2]  243:        Py:=WhereY;
[000]  244:
[BL8]  245:        IF ( Px > 1 ) OR ( rueck = 0 ) THEN B
          EGIN
[LME]  246:          FOR i:=Position(Px - rueck, Py) TO
          Position(79, Py) DO
[HN2]  247:            IF mono
[5Z9]  248:              THEN MonoScr[i]:=MonoScr[Succ(i
          )]
[WFB]  249:              ELSE WriteColor(ReadColor(Succ(
          i)), i);
[HN2]  250:          IF mono
[TNF]  251:            THEN WITH MonoScr[Position(80, Py
          )] DO BEGIN ch:=' '; at:=0; END
[1P8]  252:            ELSE WColor(' ', 0, Position(80,
          Py));
[MB8]  253:          GotoXY(WhereX - rueck, WhereY);
[TM0]  254:        END;
[TM0]  255:      END;
[SMB]  256:    {-----------------------------------------
          --------------------------------}
[000]  257:
[916]  258:    PROCEDURE InsertChar;
```

```
[000]  259:
[NH6]  260:      VAR i, Px, Py, ofs : Integer;
[000]  261:
[B21]  262:      BEGIN
[6V4]  263:        geaendert:= TRUE;
[EW2]  264:        Px:=WhereX;
[KW2]  265:        Py:=WhereY;
[GD2]  266:        IF InsMod
[6GF]  267:          THEN FOR i:=Position(80, Py) DOWNTO
         Position(Succ(Px), Py) DO
[HN2]  268:                 IF mono
[3Z9]  269:                   THEN MonoScr[i]:=MonoScr[P
         red(i)]
[RFB]  270:                   ELSE WriteColor(ReadColor(
         Pred(i)), i);
[L46]  271:        ofs:=Position(Px, Py);
[HN2]  272:        IF mono
[ZY1]  273:          THEN BEGIN
[BX7]  274:                 MonoScr[ofs].ch:=taste;
[LE8]  275:                 MonoScr[ofs].at:=Attribut;
[DD0]  276:               END
[EM9]  277:          ELSE WColor(taste, Attribut, ofs);
[TG3]  278:        IF WhereX < 80
[YU9]  279:          THEN GotoXY(Succ(WhereX), WhereY)
[SMB]  280:          ELSE IF ofs = 2000 THEN GotoXY(1, S
         ucc(WhereY));
[TM0]  281:      END;
[SMB]  282:   {-------------------------------------
         -------------------------------}
[000]  283:
[4PK]  284:    PROCEDURE Disp ( s : Str80; color : Byte;
         invers, hell : Boolean);
[000]  285:
[12K]  286:      { gibt an der aktuellen CursorPosition
         einen String aus }
[000]  287:
[PD7]  288:      VAR i, ofs, Px, Py      : Integer;
[X64]  289:          atr                 : Byte;
[000]  290:
[B21]  291:      BEGIN
[HN2]  292:        IF mono
[K24]  293:          THEN IF invers
[203]  294:                 THEN atr:=112
[GU2]  295:                 ELSE IF hell
[4S2]  296:                        THEN atr:=10
[6K2]  297:                        ELSE atr:=2
[YY1]  298:          ELSE BEGIN
[TX2]  299:                 atr:= color;
[5T8]  300:                 IF invers THEN atr:=atr + in
         v;
[BB8]  301:                 IF hell Then atr:=atr + high
         ;
[TM0]  302:               END;
[EW2]  303:        Px:=WhereX;
[KW2]  304:        Py:=WhereY;
[L46]  305:        ofs:=Position(Px, Py);
[0A7]  306:        FOR i:=1 TO Length(s) DO BEGIN
[HN2]  307:          IF mono
[WXE]  308:            THEN WITH MonoScr[ofs] DO BEGIN
         ch:=s[i]; at:=atr; END
[X07]  309:            ELSE WColor(s[i], atr, ofs);
[WH4]  310:          ofs:=Succ(ofs);
[TM0]  311:        END;
[3D6]  312:        GotoXY(1, Succ(WhereY));
[TM0]  313:      END;
[SMB]  314:   {-------------------------------------
         -------------------------------}
```

**Listing 3. Die Datei »screen.inc« unterscheidet zwischen Hercules- und CGA-Grafikkarte**

```
[000]    0:
[FKA]    1: (*****************************************
           ******************************)
[B48]    2: (*                                       *)
[D59]    3: (*                      T A S T . I N C  *)
[B48]    4: (*                                       *)
[F9G]    5: (*  Autor          :   Torsten Knappe Jahns
           tr. 4   4790 Paderborn                     *)
[B48]    6: (*                                       *)
[P1A]    7: (*  Start          :   22.07.87            *)
[AGB]    8: (*  akt. Stand     :   09.09.87            *)
[B48]    9: (*                                       *)
[68N]   10: (*  Prgm. Beschr.  :  Includedatei mit Tast
           aturroutinen fuer MASKED.PAS  *)
[B48]   11: (*                                       *)
[FKA]   12: (*****************************************
           ******************************)
[000]   13:
[000]   14:
[3H7]   15:    PROCEDURE BelegFktTasten;
[000]   16:
[KEN]   17:      { Belegt die Funktionstasten SF1 - AF1
           0 mit Rahmen- und Fuellsymbole }
[Z4K]   18:      { Alle anderen Tasten werden mit 0 ini
           tialisiert.                    }
[000]   19:
[LH3]   20:      VAR i : Codes;
[000]   21:
[B21]   22:      BEGIN
[U9A]   23:        FOR i:=up TO SF10 DO FktTastenTab[i]
           :=0;
[WP6]   24:          FktTastenTab[SF1 ]:=201;
[HT6]   25:          FktTastenTab[SF2 ]:=187;
[NP6]   26:          FktTastenTab[SF3 ]:=200;
[ST6]   27:          FktTastenTab[SF4 ]:=188;
[AT6]   28:          FktTastenTab[SF5 ]:=186;
[SR6]   29:          FktTastenTab[SF6 ]:=205;
[HR6]   30:          FktTastenTab[SF7 ]:=204;
[3T6]   31:          FktTastenTab[SF8 ]:=185;
[3R6]   32:          FktTastenTab[SF9 ]:=202;
[UR6]   33:          FktTastenTab[SF10]:=203;
[000]   34:
[LM6]   35:          FktTastenTab[AF1 ]:=206;
[SN6]   36:          FktTastenTab[AF2 ]:=176;
[ZN6]   37:          FktTastenTab[AF3 ]:=177;
[6P6]   38:          FktTastenTab[AF4 ]:=178;
[EN6]   39:          FktTastenTab[AF5 ]:=219;
[FK6]   40:          FktTastenTab[AF6 ]:= 64;
[MM6]   41:          FktTastenTab[AF7 ]:=116;
[AL6]   42:          FktTastenTab[AF8 ]:=110;
[HN6]   43:          FktTastenTab[AF9 ]:=175;
[HK6]   44:          FktTastenTab[AF10]:= 62;
[TM0]   45:        END;
[SMB]   46:     {-------------------------------------
           ---------------------------------}
[000]   47:
[A5D]   48:     PROCEDURE Get (VAR Taste : Char; VAR Bed
           : Codes);
[000]   49:
[5S7]   50:       { liefert Tastaturcode }
[000]   51:
[B21]   52:       BEGIN
[855]   53:         Read(kbd, Taste);
[2PE]   54:         IF (Taste = #27) AND KeyPressed
                  { dann ScanCode }
[ZY1]   55:            THEN BEGIN
[855]   56:                   Read(kbd, Taste);
[5P4]   57:                   CASE Ord(Taste) OF
[WV3]   58:                        115 : Bed:=Cleft;
[R54]   59:                         75 : Bed:=left;
[B34]   60:                        116 : Bed:=Cright;
[9H4]   61:                         77 : Bed:=right;
[YL3]   62:                        160 : Bed:=Cup;
[CY2]   63:                         72 : Bed:=up;
[E04]   64:                        164 : Bed:=Cdown;
[F74]   65:                         80 : Bed:=down;
[UZ3]   66:                         73 : Bed:=PgUp;
[VB4]   67:                         81 : Bed:=PgDown;
[MB4]   68:                         71 : Bed:=ToHome;
[N74]   69:                         79 : Bed:=ToEnd;
[0T3]   70:                    82, 165 : Bed:=ins;
[KS3]   71:                    83, 166 : Bed:=del;
[FK2]   72:                         59 : Bed:=F1;
[2K2]   73:                         60 : Bed:=F2;
[8K2]   74:                         61 : Bed:=F3;
[EK2]   75:                         62 : Bed:=F4;
[MK2]   76:                         63 : Bed:=F5;
[UK2]   77:                         64 : Bed:=F6;
[0L2]   78:                         65 : Bed:=F7;
[6L2]   79:                         66 : Bed:=F8;
[CL2]   80:                         67 : Bed:=F9;
[TV2]   81:                         68 : Bed:=F10;
[LZ2]   82:                         84 : Bed:=SF1;
[UZ2]   83:                         85 : Bed:=SF2;
[103]   84:                         86 : Bed:=SF3;
[803]   85:                         87 : Bed:=SF4;
[F03]   86:                         88 : Bed:=SF5;
[P03]   87:                         89 : Bed:=SF6;
[A03]   88:                         90 : Bed:=SF7;
[H03]   89:                         91 : Bed:=SF8;
[S03]   90:                         92 : Bed:=SF9;
[DA3]   91:                         93 : Bed:=SF10;
[4Y2]   92:                         94 : Bed:=CF1;
[BY2]   93:                         95 : Bed:=CF2;
[KY2]   94:                         96 : Bed:=CF3;
[TY2]   95:                         97 : Bed:=CF4;
[0Z2]   96:                         98 : Bed:=CF5;
[7Z2]   97:                         99 : Bed:=CF6;
[H13]   98:                        100 : Bed:=CF7;
```

```
[U13]   99:                        101 : Bed:=CF8;
[323]  100:                        102 : Bed:=CF9;
[9Y2]  101:                        103 : Bed:=CF10;
[H03]  102:                        104 : Bed:=AF1;
[U03]  103:                        105 : Bed:=AF2;
[313]  104:                        106 : Bed:=AF3;
[C13]  105:                        107 : Bed:=AF4;
[N13]  106:                        108 : Bed:=AF5;
[Y13]  107:                        109 : Bed:=AF6;
[B13]  108:                        110 : Bed:=AF7;
[M13]  109:                        111 : Bed:=AF8;
[X13]  110:                        112 : Bed:=AF9;
[3Y2]  111:                        113 : Bed:=AF10;
[D46]  112:                    ELSE       Bed:=Sonstige;
[TM0]  113:                    END;
[DD0]  114:                END
[6TA]  115:            ELSE BEGIN
                  { normaler Code }
[5P4]  116:                CASE Ord(Taste) OF
[YY2]  117:                  13  : Bed:=cr;
[1G3]  118:                  27  : Bed:=Esc;
[WV2]  119:                   8  : Bed:=bs;
[915]  120:                  ELSE  Bed:=Sonstige;
[TM0]  121:                END;
[TM0]  122:             END;
[TM0]  123:        END;
[SMB]  124:    {-----------------------------------
          ---------------------------------}
[000]  125:
[T86]  126:    PROCEDURE GetFileName;
[000]  127:
[PTF]  128:      { liest einen Dateinamen ( ohne extensi
          on ) ein }
[000]  129:
[857]  130:      VAR  Px, Py, ofs : Integer;
[Y13]  131:           atr         : Byte;
[000]  132:
[B21]  133:      BEGIN
[3D3]  134:        FileName:='';
[X42]  135:        Repeat
[6Y4]  136:          Get(Taste, Code);
[MY6]  137:          Px:=WhereX; Py:=WhereY;
[L46]  138:          ofs:=Position(Px, Py);
[RN2]  139:          CASE Code OF
[GP4]  140:            Sonstige : BEGIN
[EN5]  141:                         Taste:=UpCase(Taste)
          ;
[VS8]  142:                         IF ( ( Taste IN ['A'
          ..'Z', '_', '0'..'9'] ) AND
[YR8]  143:                              ( Length(FileNa
          me) < 8 )) THEN BEGIN
[EF7]  144:                           FileName:=FileNam
          e + Taste;
[HN2]  145:                           IF mono
[2P8]  146:                             THEN WITH MonoS
          cr[ofs] DO BEGIN
[4S5]  147:                                  ch:=Tast
          e; at:=MHigh;
[DD0]  148:                                END
[B7B]  149:                             ELSE WColor(tas
          te, white + high, ofs);
[1R7]  150:                           GotoXY(Succ(Where
          X), WhereY);
[TM0]  151:                         END;
[TM0]  152:                       END;
[6V2]  153:              bs       : BEGIN
[4E8]  154:                         IF Length(Filename)
          > 0 THEN BEGIN
[5R7]  155:                           GotoXY(Pred(WhereX
          ), WhereY);
[N98]  156:                           ofs:=Position(Wher
          eX, WhereY);
[HN2]  157:                           IF mono
[XP6]  158:                             THEN MonoScr[ofs
          ].ch:=' '
[7N9]  159:                             ELSE WColor(' ',
           GetCharAt(ofs), ofs);
[VUB]  160:                           Delete(Filename, L
          ength(Filename), 1);
[TM0]  161:                         END;
[TM0]  162:                       END;
[TM0]  163:          END;
[GFF]  164:        UNTIL ( Code = esc ) OR (( Code = cr
          ) AND (Length(FileName) > 0 ));
[TM0]  165:      END;
[SMB]  166:    {-----------------------------------
          ---------------------------------}
[000]  167:
```

**Listing 4. Mit »tast.inc« werden die Tastatureingaben überprüft**

```
[FKA]    0: (*****************************************
           ******************************)
[B48]    1: (*                                       *)
[LS8]    2: (*                        U T Y . I N C  *)
[B48]    3: (*                                       *)
[F9G]    4: (*  Autor          :   Torsten Knappe Jahns
           tr. 4   4790 Paderborn                     *)
[B48]    5: (*                                       *)
[P1A]    6: (*  Start          :   22.07.87           *)
[3GB]    7: (*  akt. Stand     :   22.09.87           *)
[B48]    8: (*                                       *)
[NNG]    9: (*  Prgm. Beschr.  :  Includedatei fuer MAS
           KEDIT.PAS                                  *)
[B48]   10: (*                                       *)
[FKA]   11: (*****************************************
           ******************************)
[000]   12:
[5H5]   13:   PROCEDURE Parameter;
[000]   14:
[WUA]   15:     { zeigt wichtige Parameter an }
[000]   16:
[3C6]   17:     VAR  c, d          : String[2];
[7E5]   18:          Farbe         : String[7];
[000]   19:
[B21]   20:     BEGIN
[DR2]   21:       SavePage;
[B13]   22:       CursorOff;
[4B2]   23:       Str(Px,c);
[2N8]   24:       IF Length(c) = 1 THEN Insert(' ',c, 1
          );
[AB2]   25:       Str(Py,d);
[7N8]   26:       IF Length(d) = 1 THEN Insert(' ',d, 1
          );
[593]   27:       GotoXY(1, 15);
[0T8]   28:       Disp('################ P A R A M E T
          E R ##################',
[DN4]   29:            black , True, True);
[7KE]   30:       Disp('akt. Cursor Position :  Zeile:
          ' + d + '    Spalte: '
[LB7]   31:            + c + '    ', white, False, Tr
          ue);
[S08]   32:       Disp('################################
          ######################',
[1K4]   33:            black, True, True);
[V53]   34:       IF NOT InsMod
[ZEE]   35:         THEN Disp('Einfügeart          :
          überschreiben                ',
[VF4]   36:              white, False, True)
[NRC]   37:         ELSE Disp('Einfügeart          :
          Einfügen                     ',
[7R4]   38:              white, False, True);
[S08]   39:       Disp('################################
          ######################',
[1K4]   40:            black, True, True);
[1R7]   41:       IF HighMod THEN c:='+' ELSE c:='-';
[K0F]   42:       Disp('  ' + c + ' Hell
                    ', white, False, True);
[H48]   43:       IF BlinkMod THEN c:='+' ELSE c:='-';
[WRF]   44:       Disp('  ' + c + ' Blinkend
                    ', white, False, True);
[D37]   45:       IF InvMod THEN c:='+' ELSE c:='-';
[ZLF]   46:       Disp('  ' + c + ' Invers
                    ', white, False, True);
[HN2]   47:       IF mono
[ZY1]   48:         THEN BEGIN
[A48]   49:                IF UnderMod THEN c:='+' ELSE
           c:='-';
[FDA]   50:              Disp('  ' + c + ' Unterstrei
          chen                         ',
[7R4]   51:                   white, False, True);
[DD0]   52:            END
[YY1]   53:         ELSE BEGIN
[994]   54:                CASE AktColor OF
[LF5]   55:                  0   : Farbe:='Schwarz';
[5D4]   56:                  1   : Farbe:='Blau   ';
[VE4]   57:                  2   : Farbe:='Grün   ';
[925]   58:                  3   : Farbe:='Türkis ';
[1C4]   59:                  4   : Farbe:='Rot    ';
[VA5]   60:                  5   : Farbe:='Magenta';
[KD4]   61:                  6   : Farbe:='Gelb   ';
[YS4]   62:                  7   : Farbe:='Weiß   ';
[TM0]   63:                END;
[HDC]   64:                Disp('aktuelle Farbe       :
          ' + Farbe  + '                ',
[BX4]   65:                     white, False, True );
[TM0]   66:            END;
[PLB]   67:       Disp('################################
          weiter mit < Esc > ###',
```

```
[1K4]   68:              black, True, True);
[D17]   69:         GotoXY(77, Pred(WhereY));
[XG1]   70:         REPEAT
[6Y4]   71:           Get(Taste, Code);
[2U3]   72:         UNTIL Code = Esc;
[R35]   73:         WriteSavedPage;
[AW2]   74:         CursorON;
[P42]   75:         Code:=F9;
[TM0]   76:       END;
[SMB]   77:    {-----------------------------------
          --------------------------------}
[000]   78:
[2T5]   79:    PROCEDURE LoadScreen;
[000]   80:
[VVC]   81:      { laedt einen Bildschirm von der disk }
[000]   82:
[A45]   83:      VAR  ok     : Boolean;
[Y24]   84:           scr    : ScrTyp;
[L04]   85:           i      : Integer;
[000]   86:
[B21]   87:      BEGIN
[DR2]   88:        SavePage;
[B03]   89:        CursorOFF;
[XG1]   90:        REPEAT
[SS2]   91:          GotoXY(1,1);
[NF9]   92:          Disp('##############################
          ####################################',
[1K4]   93:               black, True, True);
[BBE]   94:          Disp('Laden eines Bildschirminhalte
          s                                 ',
[7R4]   95:               white, False, True);
[NF9]   96:          Disp('##############################
          ####################################',
[1K4]   97:               black, True, True);
[9Y9]   98:          Disp('Dateiname :
                                            ',
[7R4]   99:               white, False, True);
[ERB]  100:          Disp('##############################
          ################< Esc > = Abbruch ###',
[1K4]  101:               black, True, True);
[F83]  102:          GotoXY(15, 4);
[AW2]  103:          CursorON;
[LH3]  104:          GetFileName;
[B03]  105:          CursorOFF;
[478]  106:          IF Length(FileName) > 0 THEN BEGIN
[DGB]  107:            IF NOT (FileExists(Filename + '.S
          CR')) THEN BEGIN
[ZZ2]  108:              GotoXY(1, 7);
[NF9]  109:              Disp('##########################
          ######################################',
[1K4]  110:                   black, True, True);
[0U8]  111:              Disp('
                                                ',
[1K4]  112:                   black, True, True);
[TWE]  113:              Disp('A C H T U N G !!!!  Die D
          atei existiert nicht !!!!!              ',
[CM4]  114:                   white, True, True);
[7SF]  115:              Disp('         Geben Sie bitte
          einen neuen Namen ein !                 ',
[CM4]  116:                   white, True, True);
[0U8]  117:              Disp('
                                                ',
[1K4]  118:                   black, True, True);
[NF9]  119:              Disp('##########################
          ######################################',
[1K4]  120:                   black, True, True);
[TM0]  121:            END;
[TM0]  122:          END;
[V9D]  123:        UNTIL (FileExists(Filename + '.SCR'))
           OR (Code = Esc);
[52E]  124:        IF NOT (Code = Esc) THEN GetScreen(sc
          r, ok, FileName);
[495]  125:        IF ok AND NOT (Code = Esc)
[WV2]  126:          THEN IF mono
[XR6]  127:                 THEN SavedMonoScr:=scr
[1K7]  128:                 ELSE SavedColorScr:=scr;
[R35]  129:        WriteSavedPage;
[AW2]  130:        CursorON;
[P42]  131:        Code:=F2;
[TM0]  132:      END;
[SMB]  133:    {-----------------------------------
          --------------------------------}
[000]  134:
[GS7]  135:    PROCEDURE PrgmBeschreibung;
[000]  136:
[UFG]  137:      { laedt den Bildschirm der das Programm
          beschreibt }
[000]  138:
[4L5]  139:      VAR ok          : Boolean;
[0Y3]  140:          scr         : ScrTyp;
[KW3]  141:          taste       : Char;
[0S3]  142:          Code        : Codes;
[U04]  143:          i           : Integer;
[000]  144:
[B21]  145:      BEGIN
[B03]  146:        CursorOFF;
```

```
[MR7]  147:        GetScreen(scr, ok, 'Beschr');
[5N3]  148:        IF  ok THEN BEGIN
[HN2]  149:          IF mono
[FM4]  150:            THEN MonoScr:=scr
[T5C]  151:            ELSE FOR i:=1 TO 2000 DO WriteCol
          or(scr[i].w, i);
[XG1]  152:          REPEAT
[6Y4]  153:            Get(Taste, Code);
[XE3]  154:          UNTIL Code  = Esc;
[TM0]  155:        END;
[AW2]  156:        CursorON;
[TM0]  157:      END;
[SMB]  158:    {-----------------------------------
          --------------------------------}
[000]  159:
[DV5]  160:    PROCEDURE HelpScreen;
[000]  161:
[AZ9]  162:      { laedt die Hilfsbildschirme }
[000]  163:
[T05]  164:      VAR help    : ScrTyp;
[EV3]  165:          ok      : Boolean;
[PR3]  166:          i       : Integer;
[000]  167:
[B21]  168:      BEGIN
[DR2]  169:        SavePage;
[B03]  170:        CursorOFF;
[2H7]  171:        GetScreen(help, ok, 'HELP1');
[ZU3]  172:        IF ok THEN BEGIN
[HN2]  173:          IF mono
[WS4]  174:            THEN MonoScr:=help
[VZB]  175:            ELSE FOR i:=1 TO 2000 DO WriteCol
          or(help[i].w, i);
[FL3]  176:          GotoXY(80, 25);
[XG1]  177:          REPEAT
[6Y4]  178:            Get(Taste, Code);
[Y36]  179:          UNTIL Code IN [ Esc, cr ];
[896]  180:          IF Code = cr THEN BEGIN
[4H7]  181:            GetScreen(help, ok, 'HELP2');
[ZU3]  182:            IF ok THEN BEGIN
[HN2]  183:              IF mono
[WS4]  184:                THEN MonoScr:=help
[VZB]  185:                ELSE FOR i:=1 TO 2000 DO Writ
          eColor(help[i].w, i);
[FL3]  186:              GotoXY(80, 25);
[XG1]  187:              REPEAT
[6Y4]  188:                Get(Taste, Code);
[6W4]  189:              UNTIL Code IN [ Esc];
[TM0]  190:            END;
[TM0]  191:          END;
[TM0]  192:        END;
[R35]  193:        WriteSavedPage;
[AW2]  194:        CursorON;
[182]  195:        Code:=F10;
[TM0]  196:      END;
[SMB]  197:    {-----------------------------------
          --------------------------------}
[000]  198:
[CU6]  199:    PROCEDURE Vorstellung;
[000]  200:
[TH9]  201:      { laedt den Titelbildschirm }
[000]  202:
[KH5]  203:      VAR ok          : Boolean;
[WU3]  204:          scr         : ScrTyp;
[4B4]  205:          taste       : Char;
[2X3]  206:          Code        : Codes;
[WV3]  207:          i           : Integer;
[000]  208:
[B21]  209:      BEGIN
[B03]  210:        CursorOFF;
[0S6]  211:        GetScreen(scr, ok, 'VOR');
[5N3]  212:        IF  ok THEN BEGIN
[HN2]  213:          IF mono
[FM4]  214:            THEN MonoScr:=scr
[T5C]  215:            ELSE FOR i:=1 TO 2000 DO WriteCol
          or(scr[i].w, i);
[XG1]  216:          REPEAT
[6Y4]  217:            Get(Taste, Code);
[EX6]  218:          UNTIL Code  IN [ esc, cr, F10 ];
[TM0]  219:        END;
[RN2]  220:        CASE Code OF
[BT1]  221:          F10 : BEGIN
[NF2]  222:                  ClrScr;
[R83]  223:                  HelpScreen;
[TM0]  224:                END;
[VT6]  225:          CR  : PrgmBeschreibung;
[C71]  226:          ELSE ;
[TM0]  227:        END;
[AW2]  228:        CursorON;
[TM0]  229:      END;
[SMB]  230:    {-----------------------------------
          --------------------------------}
```

**Listing 5. »uty.inc« enthält die Prozeduren, die die Hilfsbildschirme lädt**

# Verschiedene Cursor wie bei GEM

**Wenn Sie Anwendungen schreiben, bekommen die meisten Applikationen erst durch den Einsatz der Maus ihren letzten Schliff. Mit diesem Utility können Sie Ihre Programme interessanter und attraktiver gestalten.**

Sicher kennen Sie das charakteristische Stundenglas und den schrägen Pfeil der GEM-Applikationen. Mit unseren Grafik-Utilities können Sie diese Cursorformen und noch einige mehr in eigene Programme einbauen.

Interessant sind für Sie, wenn Sie eine EGA-Karte besitzen, vor allem die Includedateien. Zusätzlich sind noch zwei Demoprogramme »direktde.pas« (Listing 4) und »mausdemo.pas« (Listing 5) abgedruckt, an denen Sie sehen können, wie man die Routinen in Programme einbindet. Die Typvereinbarungen befinden sich in der Datei »typ.ega« (Listing 1). Diese Datei sollte mit der Compileranweisung »{$I »Dateiname«}« vor dem Programm »ega.inc« aufgerufen werden, weil in ihr die Typvereinbarungen für »ega.inc« enthalten sind. Erst in der Datei »ega.inc« (Listing 2) befinden sich nämlich die Routinen zur Bedienung des EGA-Grafikadapters im hochauflösendem Grafikmodus, die das direkte Schreiben in den Bildschirm erlauben.

Auch die Include-Datei »mouse.inc« (Listing 3) besitzt durchaus professionellen Charakter. Neben der Sanduhr und dem schrägen Pfeil können sie als Cursorsymbole noch die Hand, das Kreuz und den Haken aufrufen. Das sind aber noch nicht alle möglichen Cursorformen! Dazu kommen noch Prozeduren, die Maustasteneingaben überprüfen und ausführen. Eine Anzeige informiert sie über die aktuellen Positionen des Maus-Cursors.

Voraussetzung sind eine EGA-Karte und ein Colormonitor. Listing 4 und 5 sind aber ausreichend dokumentiert, so daß sich die Anpassung für eine EGA-Karte mit Monochrommonitor sehr einfach gestaltet.

Diese Utility leistet Ihnen bei der Gestaltung eigener Programme sicher wertvolle Hilfe. Durch das Variieren der Maus-Cursor wird der Aufbau eines Programms für einen Anwender verständlicher. Auch brauchen dann Ihre selbstgeschriebenen Applikationen den Vergleich mit kommerzieller Software nicht mehr zu scheuen.

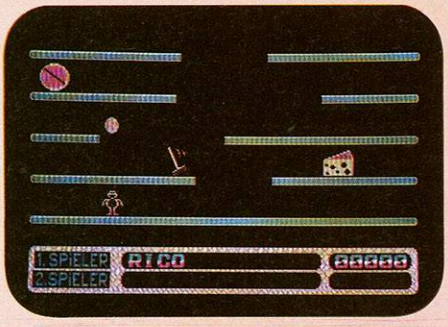
(Torsten Knappe/gh)

```
[FKA]    0: (*****************************************
            ******************************)
[B48]    1: (*                                       *)
[359]    2: (*                          T Y P . E G A
                                                     *)
[B48]    3: (*                                       *)
[F9G]    4: (*  Autor          :   Torsten Knappe Jahns
           tr. 4   4790 Paderborn                    *)
[B48]    5: (*                                       *)
[SOA]    6: (*  Start          :   10.11.86
                                                     *)
[2GB]    7: (*  akt. Stand     :   21.09.87
                                                     *)
[B48]    8: (*                                       *)
[KLM]    9: (*  Prgm. Beschr.  :  Includedatei mit Typv
           ereinbarungen fuer EGA.INC                *)
[B48]   10: (*                                       *)
[FKA]   11: (*****************************************
            ******************************)
[000]   12:
[000]   13:
[9UH]   14:   CONST egabase = $A000;          { Startad
           resse fuer EGA - Modi 15 und 16 }
[000]   15:
[ASO]   16:   TYPE
[3B8]   17:        RegTyp  = RECORD CASE Integer OF
[X6G]   18:                            1 : (ax, bx, cx, dx, bp,
           si, di, ds, es, flags : Integer);
[FHC]   19:                            2 : (al, ah, bl, bh, cl,
           ch, dl, dh            : Byte   );
[TMO]   20:                    END;
[000]   21:
[N16]   22:        Strg255 = String[255];
[LH5]   23:        Strg80  = String[80];
[000]   24:
[H58]   25:        FontArt = ( Font9x14, Font8x8 );
[PX4]   26:        FontTyp = RECORD
[RW4]   27:                    Art           : FontArt;
[2BE]   28:                    Segm, Ofs     : Integer;
             { Segment-, Offsetadresse  }
[UFE]   29:                    box, pattern : Integer;
             { Box-, Patterngroesse     }
[2FE]   30:                    fak           : Integer;
             { invarianter Umrechnungs- }
[X75]   31:
```

```
             { faktor                   }
[TMO]   32:                  END;
[000]   33:
[HK3]   34:      PTyp    = RECORD
[1EB]   35:                    x, y      : Integer;
             { Koordinaten              }
[ER4]   36:
             { x : 0 - 639, y : 0 - 349  }
[TGE]   37:                    funktion : Byte   ;
             { 0 : keine Aenderung      }
[CU6]   38:
             { 1 : UND  mit gel. Daten   }
[AW6]   39:
             { 2 : ODER mit gel. Daten   }
[NV6]   40:
             { 3 : XOR  mit gel. Daten   }
[TCA]   41:                    color     : Byte   ;
             { Farbe 0 - 15             }
[TMO]   42:                  END;
[000]   43:
[HX3]   44:      ChTyp   = RECORD
[FCB]   45:                    x, y      : Integer;
             { Koordinaten :            }
[OH4]   46:
             {                 x : 0 - 79    }
[046]   47:
             { Font9x14 :   y : 0 - 24    }
[Z26]   48:
             { Font8x8  :   y : 0 - 34    }
[76D]   49:                    inv       : Boolean;
             { inverse Darstellung ?    }
[7LA]   50:                    color  .  : Integer;
             { Farbe 0 - 15             }
[RR2]   51:                    CASE Byte OF
[9CA]   52:                     1 : ( c : Byte ) ;
             { Zeichen  Byte            }
[3AA]   53:                     2 : ( ch: Char ) ;
             { Zeichen  ASCII           }
[TMO]   54:                  END;
[000]   55:
[5X4]   56:      StrgTyp = RECORD
[FCB]   57:                    x, y      : Integer;
             { Koordinaten :            }
[OH4]   58:
             {                 x : 0 - 79    }
[046]   59:
             { Font9x14 :   y : 0 - 24    }
[Z26]   60:
             { Font8x8  :   y : 0 - 34    }
[76D]   61:                    inv       : Boolean;
```

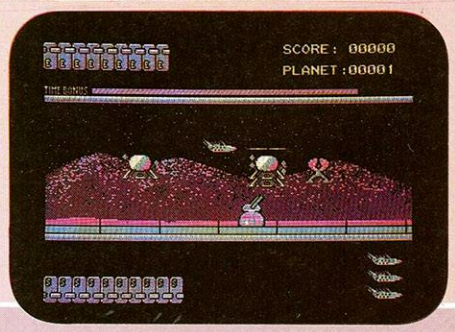
# PC-SPIELEBOX NO.2

Jetzt neu




## JACKEY

Hier erwartet Sie ein Geschicklichkeitsspiel der besonderen Art.
Sie steuern einen kleinen Roboter, der von Ebene zu Ebene springen kann. Allerdings stehen diese unter Strom, so daß Jackey versuchen muß, durch die Lücken zu springen, ohne die Ebenen von unten zu berühren. Dabei gilt es natürlich, möglichst nicht die vielen Monster zu berühren, die Jackey das Leben schwer machen. Auf der anderen Seite können Sie auf Ihrem Weg nach oben eine ganze Reihe von Boni einsammeln, die Ihren Score gehörig aufpolieren können. Programmiert von Hartmut Pfarr, dem bekannten DMV-Autor, verspricht Jackey ein echter Renner zu werden. Versäumen Sie also auf keinen Fall die aufregenden Abenteuer des kleinen Roboters Jackey.
Die Abenteuer des kleinen Jackey sind im System MS-DOS erhältlich.




## STARVISION

Ein excellentes Shoot'em up auf Ihrem PC!
Besteigen Sie Ihr Schiff und retten Sie den Mond. Außerirdische Kreaturen haben sich dort eingenistet und behindern die Aktivitäten der Menschen. Deshalb ist es jetzt Ihre Aufgabe, den Mond von diesen Wesen zu befreien. Die größte Hilfe bei Ihrer Aufgabe ist Ihr kleines Raumschiff, daß Sie im Verlauf dieses Spieles mit zahlreichen Extrafeatures ausrüsten können. Trotzdem keine einfache Aufgabe, da die Außerirdischen bereits mehrere Stützpunkte eingerichtet haben. Wenn Sie den ersten Sektor gesäubert haben, bekommen Sie einen Bonus und können sich am nächsten versuchen. Starvision verspricht somit viele Stunden spannender Unterhaltung.
Starvision ist im System MS-DOS erhältlich.

*Best.-Nr. 5 1/4" Disk. 1291*
*Best.-Nr. 3 1/2" Disk. 1292*

**49,-DM** *

# PC-SPIELEBOX NO.1

## BIMBO II

Das Geschicklichkeitsspiel für den PC überhaupt!
Wie in einem Squashcourt spielen Sie einen Ball von rechts nach links durch einen auf drei Seiten geschlossenen Bildschirm. Auf der offenen Seite müssen Sie mit Ihrem Schläger verhindern, daß der Ball ins Aus geht. Im Spielfeld selbst befinden sich unterschiedliche Hindernisse, die jedes für sich, unterschiedliche Auswirkungen auf den weiteren Spielverlauf haben. So gibt es Blöcke, die dem Spieler, sobald sie getroffen wurden, einen Extraball gutschreiben. Andere wiederum ziehen einen Ball ab, zählen Bonus, verschnellern das Spiel und und und...
Ist es Ihnen gelungen, alle Hindernisse zu beseitigen, gelangen Sie in das nächste Bild von Bimbo, das Sie wiederum mit neuen Hinderniskonstellationen konfrontiert.
**Aber Bimbo ist mehr als nur ein Spiel.** Mittels des zum Programm gehörenden Editors sind Sie in der Lage, sich genau die Bilder zusammenzusetzen, die Sie gerne spielen möchten.

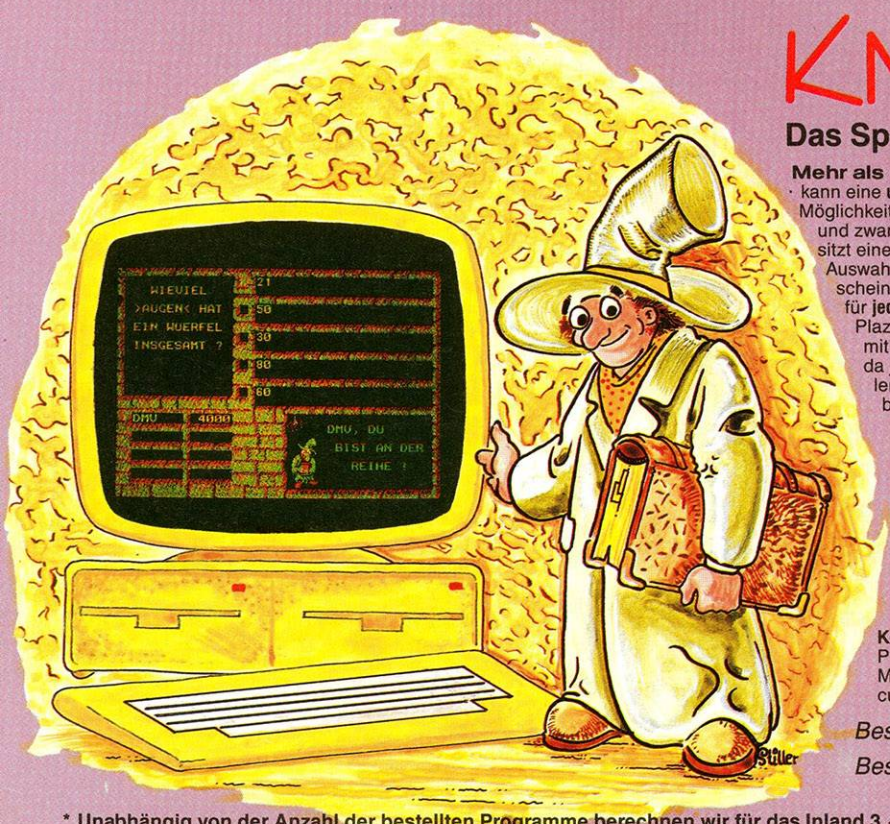
## BOUNCER

Bouncer versetzt Sie hinter das Lenkrad eines ganz besonderen Automobils. Das Bounce'O'mobil kann springen. Daß es dies nicht nur aus Spaß an der Freude tut, wird spätestens beim ersten Felsbrocken klar, der Ihnen den Weg versperrt. Nun sind Sie gefordert, überspringen Sie das Hindernis. Jedoch werden Sie nicht nur durch herumliegende Findlinge gehandicapt. So manche Brücke, die Sie auf Ihrer Fahrt überqueren müssen, ist eingestürzt, tollwütige Vögel versuchen, sich aus der Luft auf Sie zu stürzen, Straßensperren blockieren den Weg, und der rachsüchtige Pilot einer Sportmaschine stellt Ihnen nach.
Wird es Ihnen gelingen, die Staße bis zu ihrem Ende zu fahren?
Bouncer, die PC-Variante eines erfolgreichen Arcade Automaten, wurde 100% in Maschinencode geschrieben.

*Best.-Nr. 5 1/4" Disk. 129*
*Best.-Nr. 3 1/2" Disk. 185*

Für MS-DOS erhältlich

**49,-DM** *




# KNOW-PC

## Das Spiel für die ganze Familie!

**Mehr als nur ein Spiel KNOW-PC**
· kann eine **unbegrenzte** Anzahl von Fragen und Antworten verwalten · bietet die Möglichkeit, **eigene** Fragen in einem **selbstgewählten** Wissensgebiet einzugeben, und zwar mit einem **komfortablen** Editor · ist vollkommen **menuegesteuert** · besitzt eine **Supergrafik** · stellt zu jeder Frage wahlweise 5 oder 1 Antwort zur Auswahl · verhindert, daß eine **richtig** beantwortete Frage nochmals erscheint · gibt alle Fragen und Antworten sowie sämtliche Aufforderungen für **jeden** Spieler und für **jedes** Wissensgebiet mit Angabe der jeweiligen Plazierung aus · ist spielbar mit 1 – 4 Einzelspielern oder in Gruppen mit einem Vielfachen davon · kann man mit der ganzen Familie spielen, da jeder eine Mindestchance von 1 : 5 hat · muß man mit Strategie spielen, da der Beste in jedem Wissensgebiet am Ende noch Zusatzpunkte bekommt · verlangt **keinerlei** Programmierkenntnisse · wird mit 13 verschiedenen Wissensgebieten ausgeliefert · kann auch für andere Zwecke Verwendung finden, z.B. als Vokabeltrainer.

**Was ist KNOW-PC?**

KNOW-PC ist ein Frage- und Antwortspiel der Extraklasse, denn es bietet Ihnen nicht nur kurzweilige Unterhaltung, sondern ist darüber hinaus hervorragend dazu geeignet, eigene Fragen einzugeben, mit denen Sie Ihre Familie, Ihren Skatverein, Ihre Mitschüler, Kegelbrüder, Arbeitskollegen, Freunde usw. überraschen können.

**Konfiguration:**
PC XT/AT mit 512 kByte oder mehr, 1 Diskettenlaufwerk oder Festplatte, MS-DOS ab Version 2.0 oder größer, für Farbgrafik, Monochrom, Hercules oder EGA.

*Best.-Nr. 5 1/4" Disk. 161*
*Best.-Nr. 3 1/2" Disk. 162*

**49,-DM** *

*** Unabhängig von der Anzahl der bestellten Programme berechnen wir für das Inland 3,– DM bzw. für das Ausland 5,– DM Porto und Verpackung.**
**Bitte benutzen Sie die Bestellkarte**

**DMV GmbH · Postfach 250 · 3440 Eschwege**

```
                { inverse Darstellung ?      }
[7LA]   62:                      color     : Integer;
                { Farbe 0 - 15               }
[RR2]   63:                      CASE Byte OF
[X4C]   64:                       1 : ( c : ARRAY[0..80]
          OF Byte ) ; { Zeichen  Byte      }
[6NA]   65:                       2 : ( ch: Strg80 ) ;
                    { Zeichen  ASCII     }
[TMO]   66:                  END;
```

**Listing 1. Die Datei »typ.ega« enthält die Typvereinbarungen**

```
[7FB]    0: (*****************************************
           ***********************************)
[RW8]    1: (*                                       *)
[BH9]    2: (*                              E G A . I N C
                                            *)
[RW8]    3: (*                                       *)
[V1H]    4: (*  Autor          :   Torsten Knappe Jahns
           tr. 4   4790 Paderborn                     *)
[RW8]    5: (*                                       *)
[4TA]    6: (*  Start          :   10.11.86
                                            *)
[F8C]    7: (*  akt. Stand     :   22.09.87
                                            *)
[RW8]    8: (*                                       *)
[17N]    9: (*  Prgm. Beschr.  :  Includedatei mit Rout
           inen zur Bedienung des EGA-                *)
[ZBL]   10: (*                    Grafikadapters im hoc
           haufloesenden Grafikmodus                  *)
[RW8]   11: (*                                       *)
[7FB]   12: (*****************************************
           ***********************************)
[000]   13:
[000]   14:
[PZB]   15: (************************    B I O S - Call
           s    ******************************)
[000]   16:
[CA8]   17: FUNCTION KeyPressed : Boolean;
[000]   18:
[9UM]   19:   { ersetzt die Standard Prozedur KeyPresse
           d, die nicht sauber im }
[KCB]   20:   { Grafikmodus laeuft.
                                 }
[000]   21:
[8W5]   22:   VAR Reg       : RegTyp;
[243]   23:       ch        : Char;
[000]   24:
[B21]   25:   BEGIN
[43E]   26:     Reg.ah:=$0C;                  { Flush Bu
           ffer, Read Keyboard      }
[1T2]   27:     Reg.al:=$07;
[LP3]   28:     Intr($21, Reg);
[8C4]   29:     ch:= Char(Reg.al);
[4R6]   30:     KeyPressed:=Reg.al <> 0;
[TMO]   31:   END;
[PXC]   32: {-------------------------------------------
           -----------------------------------}
[000]   33:
[ZN9]   34: FUNCTION GetVideoMode : Integer;
[000]   35:
[HAD]   36:   { Gibt den Videomode zurueck. Wichtige Mo
           di :   }
[US7]   37:   {   3   : CGA        TextMode
                  }
[7Y9]   38:   {   7   : monochrome TextMode
                  }
[K7A]   39:   {   15  : Grafik EGA monochrom
                  }
[1V8]   40:   {   16  : Grafik EGA color
                  }
[000]   41:
[OX4]   42:   VAR Reg : RegTyp;
[000]   43:
[B21]   44:   BEGIN
[WD3]   45:     Reg.ax:=$0F00;
[A83]   46:     Intr($10,Reg);
[BG8]   47:     GetVideoMode:=Reg.ax AND $00FF;
[TMO]   48:   END;
[PXC]   49: {-------------------------------------------
           -----------------------------------}
```

```
[000]   50:
[VYB]   51: PROCEDURE SetVideoMode ( mode : Integer );
[000]   52:
[3DK]   53:   { Schaltet in den gewuenschten Modus um.
           Wichtige Modi :   }
[H69]   54:   {   3   : CGA        TextMode
                  }
[YBB]   55:   {   7   : monochrome TextMode
                  }
[9NB]   56:   {   15  : Grafik EGA monochrom
                  }
[S8A]   57:   {   16  : Grafik EGA color
                  }
[A9H]   58:   { Beim umschalten wird der Bildspeicher g
           eloescht.        }
[000]   59:
[OX4]   60:   VAR Reg : RegTyp;
[000]   61:
[B21]   62:   BEGIN
[ER7]   63:     IF mode IN [0..7, 13..16] THEN BEGIN
[6M3]   64:       Reg.ax:=mode;
[A83]   65:       Intr($10,Reg);
[TMO]   66:     END;
[TMO]   67:   END;
[PXC]   68: {-------------------------------------------
           -----------------------------------}
[000]   69:
[UTG]   70: PROCEDURE SetColorTable (LogColor, PhysColo
           r : Integer);
[000]   71:
[XMG]   72:   { veraendert Farbpalette : 16 LogColor  -
           -->  64 PhysColor }
[000]   73:
[OX4]   74:   VAR Reg : RegTyp;
[000]   75:
[B21]   76:   BEGIN
[WB3]   77:     Reg.ax:=$1000;
[K2B]   78:     Reg.bx:=(PhysColor Shl 8) OR LogColor;
[A83]   79:     Intr($10,Reg);
[TMO]   80:   END;
[PXC]   81: {-------------------------------------------
           -----------------------------------}
[000]   82:
[7VC]   83: PROCEDURE GetFontpointer ( VAR Font : FontT
           yp );
[000]   84:
[MUK]   85: { Gibt die Adressen der Zeichensaetze der E
           GA-Karte zurueck. }
[7KH]   86: { Ferner wird Font vollstaendig initialisie
           rt.              }
[000]   87:
[025]   88:   VAR reg : RegTyp;
[000]   89:
[B21]   90:   BEGIN
[BD3]   91:     reg.ax:=$1130;
[AH5]   92:     IF Font.art = Font8x8
[ZY1]   93:       THEN BEGIN
[2E2]   94:              reg.bh:=3;
[P7A]   95:              Font.box:=10;
          { Boxgroesse     }
[NFC]   96:              Font.pattern:=8;
          { Patterngroesse }
[LOE]   97:              Font.fak:=800;       { Umrechnu
           ngsfaktor 80 x box }
[DDO]   98:            END
[YY1]   99:       ELSE BEGIN
[1E2]  100:              reg.bh:=2;
[RD3]  101:              Font.box:=14;
[TP4]  102:              Font.pattern:=14;
[HZ3]  103:              Font.fak:=1120;
[TMO]  104:            END;
[CT3]  105:     Intr($10, reg);
[7Y4]  106:     Font.Segm:=reg.es;
[6V4]  107:     Font.Ofs :=reg.bp;
[TMO]  108:   END;
[PXC]  109: {-------------------------------------------
           -----------------------------------}
[000]  110:
[5LK]  111: (****************   Ausgabeprozeduren die d
           irekt in den   ********************)
[FSG]  112: (****************   Bildschirmspeicher schr
           eiben          ********************)
[000]  113:
[N4L]  114: { ------ A C H T U N G  :  Die folgenden Mo
           di setzen voraus, das der   ------ }
[7MG]  115: { ------                   VideoMode 15 bzw
           . 16 selektiert ist !!!!!   ------ }
[000]  116:
[MMS]  117: { Bei den folgenden Routinen wird nur ein R
           ecord uebergeben, und nicht eine   }
[MKU]  118: { Anzahl von Einzelwerten. Dies hat den Vor
           teil, das die Aufrufzeit minimiert }
[BYN]  119: { wird, da nur eine Adresse uebergeben wird
           , und nicht mehere Werte.          }
[47T]  120: { Da ein Programm nur einmal getippt, aber
           mehrmals ausgefuehrt wird, duerfte }
```

```
[WXS]  121: { der trade-off von Tippzeit mit Laufzeit g
           erechtfertigt sein. Ferner wird     }
[4NU]  122: { durch eine expliziete Namensbezeichnung d
           ie Lesbarkeit der Prozeduraufrufe   }
[KCB]  123: { erhoeht.
                                               }
[000]  124:
[DXF]  125: { verwendete Register der EGA-Karte :
                                               }
[6F9]  126: {
                                               }
[U9K]  127: { Portadresse     I   Registerbezeichnung
           I  Zweck                            }
[PZC]  128: { ---------------I-------------------------
           I--------------------------------- }
[Z5M]  129: { PORT[$3C4]:=2   I   MapMaskRegister
           I  globaler Scchreibfilter          }
[31R]  130: { PORT[$3CE]:=0   I   Set/Reset Register
           I  in diese Ebenen wird geschrieben)
[88K]  131: { PORT[$3CE]:=1   I   Enable Set/Reset Reg.
           I  Enable / Disable                 }
[DSP]  132: { PORT[$3CE]:=3   I   DataRotate Register
           I  log. Funktionen auf Screenbyte   }
[73K]  133: { PORT[$3CE]:=8   I   BitMask Register
           I  Bitmaske setzen                  }
[HYK]  134: { PORT[$3C0]      I   Attribute Address Reg.
           I  setzt Attribute                  }
[24H]  135: { PORT[$3BA]      I            "
           I  fuer Monochrome Bildschirme      }
[P6G]  136: { PORT[$3DA]      I            "
           I  fuer Color Bildschirme           }
[000]  137:
[PXC]  138: {---------------------------------------
           ---------------------------------}
[000]  139:
[9X7]  140: PROCEDURE Pset ( VAR p : PTyp );
[000]  141:
[61B]  142:   { setzt einen Punkt am Bildschirm. }
[000]  143:
[L36]  144:   VAR Offset   : Integer;
[WC5]  145:       bitmask  : Byte;
[000]  146:
[B21]  147:   BEGIN
[AB7]  148:     bitmask:=$80 Shr (p.x AND 7);
[BL7]  149:     Offset:=(p.y * 80) + (p.x Shr 3);
[BC3]  150:
                                     { select   }
[B13]  151:     PORT[$3CE]:=0;
[ZU4]  152:     PORT[$3CF]:=p.color;
[G13]  153:     PORT[$3CE]:=1;
[FH9]  154:     PORT[$3CF]:=Akt_Speicherebenen;
[4H3]  155:
                                     { write    }
[U13]  156:     PORT[$3CE]:=3;
[EY9]  157:     PORT[$3CF]:=(p.funktion and $03) Shl 3;
[L23]  158:     PORT[$3CE]:=8;
[NC5]  159:     PORT[$3CF]:=bitmask ;
[GPC]  160:     Mem[egabase : Offset]:=Mem[egabase : Of
           fset];
[BN3]  161:
                                     { deselect }
[P2M]  162:     PORT[$3CF]:=$FF;          { erlaube wi
           eder alle Bit im Bit Mask Register }
[G13]  163:     PORT[$3CE]:=1;
[C13]  164:     PORT[$3CF]:=0;
[U13]  165:     PORT[$3CE]:=3;
[C13]  166:     PORT[$3CF]:=0;
[TMO]  167:   END;
[PXC]  168: {---------------------------------------
           ---------------------------------}
[000]  169:
[499]  170: PROCEDURE WriteChar ( VAR c : ChTyp );
[000]  171:
[VZM]  172:   { schreibt ein Zeichen das durch ChTyp un
           d Font spezifiziert ist. }
[000]  173:
[GX6]  174:   VAR     Offset, ch : Integer;
[D44]  175:           i, inv     : Byte;
[000]  176:
[B21]  177:   BEGIN
[6KA]  178:     ch:=(c.c * Font.pattern) + Font.Ofs;
[XL8]  179:     Offset:=(c.y * Font.fak) + c.x;
[NL8]  180:     IF c.inv THEN inv:=$FF ELSE inv:=0;
[E53]  181:
                                     { select   }
[B13]  182:     PORT[$3CE]:=0;
[YS4]  183:     PORT[$3CF]:=c.color;
[G13]  184:     PORT[$3CE]:=1;
[FH9]  185:     PORT[$3CF]:=Akt_Speicherebenen;
[L23]  186:     PORT[$3CE]:=8;
[4H3]  187:
                                     { write    }
[PNA]  188:     FOR i:=0 TO Pred(Font.pattern) DO BEGIN
[MMA]  189:       PORT[$3CF]:=Mem[Font.Segm: ch + i] XO
           R inv;
[GPC]  190:       Mem[egabase : Offset]:=Mem[egabase :
           Offset];
[PG5]  191:       Offset:=Offset + 80;
[TMO]  192:     END;
[BN3]  193:
                                     { deselect }
[G13]  194:     PORT[$3CE]:=1;
[C13]  195:     PORT[$3CF]:=0;
[TMO]  196:   END;
[PXC]  197: {---------------------------------------
           ---------------------------------}
[000]  198:
[YEC]  199: PROCEDURE WriteString ( VAR str : StrgTyp )
           ;
[000]  200:
[X7R]  201:   { schreibt einen String, der durch StrgTy
           p und Font spezifiziert ist }
[000]  202:
[F4A]  203:   VAR  ZeilenOfs, Offset, ch : Integer;
[V45]  204:        i, j, inv             : Byte;
[000]  205:
[B21]  206:   BEGIN
[FVA]  207:     ZeilenOfs:=(str.y * Font.fak) + str.x;
[129]  208:     IF str.inv THEN inv:=$FF ELSE inv:=0;
[BC3]  209:
                                     { select   }
[B13]  210:     PORT[$3CE]:=0;
[CE6]  211:     PORT[$3CF]:=str.color;
[G13]  212:     PORT[$3CE]:=1;
[FH9]  213:     PORT[$3CF]:=Akt_Speicherebenen;
[L23]  214:     PORT[$3CE]:=8;
[4H3]  215:
                                     { write    }
[PNA]  216:     FOR i:=0 TO Pred(Font.pattern) DO BEGIN
[EG5]  217:       Offset:=ZeilenOfs;
[RMK]  218:       FOR j:=1 TO str.c[0] DO BEGIN
           { str.c[0] : Laenge des Strings }
[L2B]  219:         ch:=(str.c[j] * Font.pattern) + Fon
           t.Ofs;
[NKA]  220:         PORT[$3CF]:=Mem[Font.Segm: ch +i] X
           OR inv;
[GLC]  221:         Mem[egabase : OffSet]:=Mem[egabase
           : OffSet];
[216]  222:         Offset:=Succ(Offset);
[TMO]  223:       END;
[037]  224:       ZeilenOfs:=ZeilenOfs + 80;
[TMO]  225:     END;
[BN3]  226:
                                     { deselect }
[G13]  227:     PORT[$3CE]:=1;
[C13]  228:     PORT[$3CF]:=0;
[TMO]  229:   END;
[PXC]  230: {---------------------------------------
           ---------------------------------}
[000]  231:
[17F]  232: FUNCTION ConvertEbenen (ebene : Integer) :
           Integer;
[000]  233:
[KUF]  234:   { Wandelt die Ebenendarstellung um.
                                               }
[48S]  235:   { ConvertEbenen wird mit den Ebenen aufge
           rufen die benutzt werden sollen.    }
[AEP]  236:   { Hierbei ist es erforderlich, die hoechs
           te Ebene zuerst anzugeben.          }
[COL]  237:   { z. B. ConvertEbenen(21) selektiert die
           Ebenen 2 und 1 ,                    }
[ZSG]  238:   {       ConvertEbenen(3210) selektiert al
           le Ebenen.                          }
[5FT]  239:   { ConvertEbenen sollte nicht direkt vom A
           nwendungsprogramm benutzt werden. }
[B21]  240:   BEGIN
[G53]  241:     CASE ebene OF
[5AC]  242:           0 : ConvertEbenen:=$01;   { nur ein
           e Ebene  }
[FB6]  243:           1 : ConvertEbenen:=$02;
[WB6]  244:           2 : ConvertEbenen:=$04;
[NC6]  245:           3 : ConvertEbenen:=$08;
[000]  246:
[XW9]  247:          10 : ConvertEbenen:=$03;    { 2 Eben
           en  }
[D06]  248:          20 : ConvertEbenen:=$05;
[A16]  249:          30 : ConvertEbenen:=$09;
[P06]  250:          21 : ConvertEbenen:=$06;
[436]  251:          31 : ConvertEbenen:=$0A;
[M36]  252:          32 : ConvertEbenen:=$0C;
[000]  253:
[0P9]  254:         210 : ConvertEbenen:=$07;   { 3 Ebene
           n  }
[VT5]  255:         310 : ConvertEbenen:=$0B;
[XT5]  256:         320 : ConvertEbenen:=$0D;
[0U5]  257:         321 : ConvertEbenen:=$0E;
[000]  258:
[N7A]  259:        3210 : ConvertEbenen:=$0F;   { 4 Ebene
           n }
[SXC]  260:        ELSE   ConvertEbenen:=$0F;   { alle Eb
           enen sonst }
[TMO]  261:     END;
```

```
[TMO]  262:   END;
[PXC]  263: {---------------------------------------
       ---------------------------------}
[000]  264:
[6HD]  265: PROCEDURE SelectSpeicherEbene (ebene : Inte
       ger);
[000]  266:
[OKN]  267:   { Nur in die selektierten Ebenen ( 0 - 3
       ) kann geschrieben werden.          }
[PLS]  268:   { SelectSpeicherEbene wird mit den Ebenen
        aufgerufen die benutzt werden      }
[HHS]  269:   { sollen. Hierbei ist es erforderlich, di
       e hoechste Ebene zuerst anzugeben. }
[4YK]  270:   { z. B. SelectSpeicherEbene(21) selektier
       t die Ebenen 2 und 1 ,             }
[HEK]  271:   {       SelectSpeicherEbene(3210) selekti
       ert alle Ebenen.                   }
[000]  272:
[B21]  273:   BEGIN
[UBP]  274:     Akt_SpeicherEbenen:=ConvertEbenen(ebene
       );      { belege globale Variable }
[9HE]  275:     PORT[$3C4]:=2;
                { Map Mask Register, er-  }
[9E7]  276:
                { laube Schreiben in      }
[YF9]  277:     PORT[$3C5]:=Akt_SpeicherEbenen;
[TMO]  278: END;
[PXC]  279: {---------------------------------------
       ---------------------------------}
[000]  280:
[ECC]  281: PROCEDURE ShowBildEbene (ebene : Integer);
[000]  282:
[MGH]  283:   { Nur die selektierte Ebenen ( 0 - 3 ) we
       rden angezeigt.                    }
[ERR]  284:   { ShowBildEbene wird mit den Ebenen aufge
       rufen die benutzt werden sollen. }
[D7P]  285:   { Hierbei ist es erforderlich, die hoechs
       te Ebene zuerst anzugeben.         }
[MEK]  286:   { z. B. ShowBildEbene(21) selektiert die
       Ebenen 2 und 1 ,                   }
[77G]  287:   {       ShowBildEbene(3210) selektiert al
       le Ebenen.                         }
[000]  288:
[MN2]  289:   VAR a : BYTE;
[000]  290:
[B21]  291:   BEGIN
[TH5]  292:    IF GetVideoMode = 16
[DMC]  293:      THEN a:=PORT[$3DA]                { Col
       or Bildschirm       }
[WUD]  294:      ELSE a:=PORT[$3BA];               { Mon
       ochrome Bildschirm  }
[G6D]  295:    PORT[$3C0]:=$12;                    { Bil
       dschirm dunkel      }
[KN9]  296:    PORT[$3C0]:=ConvertEbenen(ebene);
[43C]  297:    PORT[$3C0]:=$20;                    { Bil
       dschirm ein         }
[TMO]  298:  END;
[PXC]  299: {---------------------------------------
       ---------------------------------}
[000]  300:
[54A]  301: PROCEDURE DelEbenen (ebene : Integer);
[000]  302:
[E2S]  303:   { DelEbenen wird mit den Ebenen aufgerufe
       n die geloescht werden sollen.     }
[D7P]  304:   { Hierbei ist es erforderlich, die hoechs
       te Ebene zuerst anzugeben.         }
[X1G]  305:   { z. B. DelEbenen(21) loescht die Ebenen
       2 und 1 ,                          }
[WLE]  306:   {       DelEbenen(3210) loescht alle Eben
       en.                                }
[000]  307:
[B21]  308:   BEGIN
[8B3]  309:     PORT[$3C4] :=2;
[XC3]  310:     PORT[$3C5] :=$FF;
[B13]  311:     PORT[$3CE]:=0;
[C13]  312:     PORT[$3CF]:=0;
[G13]  313:     PORT[$3CE]:=1;
[T3M]  314:     PORT[$3CF]:=ConvertEbenen(ebene);  { au
       ch moeglich : PORT[$3CF:=color  }
[YM6]  315:                                        { co
       lor : 0 .. 15                      }
[U13]  316:     PORT[$3CE]:=3;
[Y73]  317:     PORT[$3CF]:=$00;
[L23]  318:     PORT[$3CE]:=8;
[WG3]  319:     PORT[$3CF]:=$FF;
[R3C]  320:     move(Mem[egabase : 0],Mem[egabase : 0],
       28000);
[B13]  321:     PORT[$3CE]:=0;
[C13]  322:     PORT[$3CF]:=0;
[G13]  323:     PORT[$3CE]:=1;
[C13]  324:     PORT[$3CF]:=0;
[TMO]  325:   END;
[PXC]  326: {---------------------------------------
       ---------------------------------}
[000]  327:
[BK4]  328: PROCEDURE ClrScr;
[000]  329:
[L66]  330:   BEGIN  DelEbenen(3210); END;
[PXC]  331: {---------------------------------------
       ---------------------------------}
[000]  332:
[08A]  333: PROCEDURE DelLine (y, color : Integer);
[000]  334:
[V8E]  335:   { DelLine loescht die Farbe color in Zeil
       e y. }
[000]  336:
[H77]  337:   VAR  Offset, anz  : Integer;
[000]  338:
[B21]  339:   BEGIN
[ZF6]  340:     Offset:=(y * Font.fak);
[5RC]  341:     IF Font.art = Font8x8 THEN anz:=640 ELS
       E anz:=1120;
[B13]  342:     PORT[$3CE]:=0;
[C13]  343:     PORT[$3CF]:=0;
[G13]  344:     PORT[$3CE]:=1;
[ZH4]  345:     PORT[$3CF]:=color;
[U13]  346:     PORT[$3CE]:=3;
[504]  347:     PORT[$3CF]:=1 Shl 3;
[L23]  348:     PORT[$3CE]:=8;
[WG3]  349:     PORT[$3CF]:=$FF;
[BAH]  350:     move(Mem[egabase : Offset],Mem[egabase
       : Offset], anz);
[B13]  351:     PORT[$3CE]:=0;
[C13]  352:     PORT[$3CF]:=0;
[G13]  353:     PORT[$3CE]:=1;
[C13]  354:     PORT[$3CF]:=0;
[U13]  355:     PORT[$3CE]:=3;
[C13]  356:     PORT[$3CF]:=0;
[TMO]  357:   END;
[PXC]  358: {---------------------------------------
       ---------------------------------}
[000]  359:
[216]  360: PROCEDURE InvertBild;
[000]  361:
[S1G]  362:   { Invertiert den gesamten Bildspeicherinh
       alt. }
[000]  363:
[B21]  364:   BEGIN
[8B3]  365:     PORT[$3C4] :=2;
[XC3]  366:     PORT[$3C5] :=$FF;
[B13]  367:     PORT[$3CE]:=0;
[AD3]  368:     PORT[$3CF]:=15;
[G13]  369:     PORT[$3CE]:=1;
[AD3]  370:     PORT[$3CF]:=15;
[U13]  371:     PORT[$3CE]:=3;
[W93]  372:     PORT[$3CF]:=$18;
[R3C]  373:     move(Mem[egabase : 0],Mem[egabase : 0],
       28000);
[B13]  374:     PORT[$3CE]:=0;
[C13]  375:     PORT[$3CF]:=0;
[G13]  376:     PORT[$3CE]:=1;
[C13]  377:     PORT[$3CF]:=0;
[TMO]  378:   END;
[PXC]  379: {---------------------------------------
       ---------------------------------}
```

**Listing 2. In »ega.inc« haben Sie mehrere Grafikcursor zum Ansprechen der EGA-Karte**

```
[26B]    0: (*****************************************
         ***********************************)
[4N8]    1: (*                                       *)
[WP9]    2: (*                      M O U S E . I N C
                                                     *)
[4N8]    3: (*                                       *)
[8UG]    4: (*  Autor           :   Torsten Knappe Jahns
         tr. 4   4790 Paderborn                      *)
[4N8]    5: (*                                       *)
[MHA]    6: (*  Start           :   13.11.86
                                                     *)
[WOC]    7: (*  akt. Stand      :   22.09.87
                                                     *)
[4N8]    8: (*                                       *)
[RMM]    9: (*  Prgm. Beschr.   :  Includedatei mit Unte
         rprogrammem fuer eine                       *)
[20E]   10: (*                     ( Microsoft kompatibl
         e ) Maus.                                   *)
```

```
[4N8]   11: (*
                                          *)
[26B]   12: (*****************************************
            *********************************)
[000]   13:
[YFD]   14: { ----- Alle Koordinatenangaben in Pixel !!
            ! -----}
[000]   15:
[UCA]   16: FUNCTION MouseInstalled : Boolean;
[000]   17:
[PPM]   18:   { TRUE : Maus installiert. Zusaetzlich wi
            rd die Maus rueckgesetzt }
[000]   19:
[OX4]   20:   VAR Reg : RegTyp;
[000]   21:
[B21]   22:   BEGIN
[UB3]   23:     Reg.ax:=$0000;
[2Y2]   24:     Intr(51,Reg);
[FX7]   25:     MouseInstalled:=Reg.ax = -1;
[TMO]   26:   END;
[Z7C]   27: (*-------------------------------------
            ----------------------------------*)
[000]   28:
[VM7]   29: PROCEDURE MouseCursorOn;
[000]   30:
[OX4]   31:   VAR Reg : RegTyp;
[000]   32:
[B21]   33:   BEGIN
[ZB3]   34:     Reg.ax:=$0001;
[2Y2]   35:     Intr(51,Reg);
[TMO]   36:   END;
[Z7C]   37: (*-------------------------------------
            ----------------------------------*)
[000]   38:
[YM7]   39: PROCEDURE MouseCursorOff;
[000]   40:
[OX4]   41:   VAR Reg : RegTyp;
[000]   42:
[B21]   43:   BEGIN
[4C3]   44:     Reg.ax:=$0002;
[2Y2]   45:     Intr(51,Reg);
[TMO]   46:   END;
[Z7C]   47: (*-------------------------------------
            ----------------------------------*)
[000]   48:
[NHF]   49: PROCEDURE GetMouseInfo  ( VAR x,y
                        : Integer;
[NTD]   50:                           VAR LeftKeyPresse
            d, RightKeyPressed : Boolean );
[000]   51:
[LHB]   52:   { x , y                  : Mauskoordinat
            en          }
[7PG]   53:   { LeftKeyPressed  = TRUE  : linke  Taste
            gedrueckt    }
[34H]   54:   { RightKeyPressed = TRUE  : rechte Taste
            gedrueckt    }
[000]   55:
[6D5]   56:   VAR  Reg        : RegTyp;
[000]   57:
[B21]   58:   BEGIN
[9C3]   59:     Reg.ax:=$0003;
[2Y2]   60:     Intr(51,Reg);
[GR2]   61:     x:= Reg.cx;
[KR2]   62:     y:= Reg.dx;
[DV9]   63:     LeftKeyPressed:=  (Reg.BX AND $01)<>0;
[H2A]   64:     RightKeyPressed:= (Reg.BX AND $02)<>0;
[TMO]   65:   END;
[Z7C]   66: (*-------------------------------------
            ----------------------------------*)
[000]   67:
[L8D]   68: PROCEDURE SetMouseCursorPos ( x, y : Intege
            r );
[000]   69:
[RHA]   70:   { setze Maus an Position x, y }
[000]   71:
[OX4]   72:   VAR Reg : RegTyp;
[000]   73:
[B21]   74:   BEGIN
[EC3]   75:     Reg.ax:=$0004;
[DK2]   76:     Reg.cx:=y;
[HK2]   77:     Reg.dx:=x;
[2Y2]   78:     Intr(51,Reg);
[TMO]   79:   END;
[Z7C]   80: (*-------------------------------------
            ----------------------------------*)
[000]   81:
[71F]   82: PROCEDURE GetButtonPressInfo ( button
                : Integer;
[ED8]   83:                                VAR ButtonPr
            essed  : Boolean;
[9E8]   84:                                VAR CountOfP
            ressed : Integer;
[UP6]   85:                                VAR x,y
                : Integer );
[000]   86:
[NCE]   87:   { Curorposition beim letzten click
```


**ÜBERSETZER–SOFTWARE**

Das **ÜBERSETZUNGSPROGRAMM** für PC's und KOMPATIBLE RECHNER!

Sie besitzen fremdsprachige Texte, z.B. in ENGLISCH, die Sie in eine beliebige andere Sprache übersetzen wollen! Sie haben fremdsprachige Programme, wünschen aber eine deutsche Benutzerführung! Sie besitzen diverse Public-Domain-Software, die Sie verdeutschen wollen!

Dann benötigen Sie den automatischen TEXT- und PROGRAMM-

**ÜBERSETZER**
**Speller—SpellChecker**

mit voll integriertem Wörterbuch-Editor.

Deutsche Benutzerführung, voll menügesteuert, Fenstertechnik, viele Optionen, auch während der Übersetzung. Sie sehen beide Texte gleichzeitig auf dem Bildschirm, den fremdsprachigen Text einige Zeilen im voraus.

Sie benötigen ein ganz spezielles Wörterbuch? Zum Beispiel aus dem Computer-, Elektronik-, Medizin-, Chemie-, Finanz-, Wirtschafts-Bereich und und ....

**SPSearch,** der Wörterbuchvorbereiter.

SpStrain, das universelle Programm, um aus .COM, .EXE, .OBJ ... Texte als ASCII-Texte herauszufiltern.

**SpStrain,** der universelle Textfilter.

Ein Grundwortschatz ENGLISCH wird mitgeliefert, andere sind in Vorbereitung. Jedes Wörterbuch ist leicht selbst erstellbar.

PC-Kompatible, XT, AT, ..., 512 kB, HERCULES, CGA, EGA, 2 LW oder Harddisk.
Diskettenformate: 5.25" oder 3.5".

Unsere Software ist seit 1987 auf dem PC-Markt erhältlich und wird seitdem in Firmen der unterschiedlichsten Branchen sowie im privaten Kreis erfolgreich eingesetzt.

| | |
|---|---|
| Speller-SpellChecker 1.60C | 278,– DM |
| E-Modul TransLat | 118,– DM |
| SpSearch, SpStrain | je 168,– DM |
| Paketpreis, alle vier Programme | nur 598,– DM |

Porto und Verpackung DM 7,– unabhängig von der Bestellmenge

**F.J.WILSKI**
SOFTWARE-ENTWICKLUNG
+ VERTRIEB aus BERLIN
ZWICKAUER DAMM 12
1000 BERLIN 47
RUF (030) 6617276



Der 16 MHz RENNER **286S+**

YPON

8088/286/386

4096 KB Hauptplatine m. 512 KB RAM
DSC-System (Dynamic-Speed-Controll)
CPU 80286-12, 16 MHz (LM), 0 WS
EMS-Controll-System (L.I.M.-Standart)
Automatik EMS-Installations Software
1,2 MB/360 KB Disketten-Laufwerk
42 Megabyte Festplatten-Laufwerk
Text-Datenverarbeitung, Flight-Sim
1 x Serielle, 1 x Parallele Schnittstelle
Echtzeit-Uhr-Grafikkarte (800 x 600)
Liz. BIOS, LED-Anzeige, Schaltpanel
Ergonomische DIN-Tastatur, sep. Cursor
14 Zoll Monitor NEC MultiSync GS

| Paketpreis | DM 5.498,– |
|---|---|
| Weitere YPON Modelle, ab | DM |
| XT 8088 Systeme | 1798,– |
| AT 80286 Systeme | 2498,– |
| AT 80386 Systeme | 6998,– |
| VGA-Grafikkarten, ab | 798,– |
| EGA-Grafikkarten, ab | 398,– |
| NEC MultiSync Monitore, ab | 598,– |
| QUADRAM MultiSync | 1698,– |
| EPSON/NEC, Drucker ab | 998,– |

**CALL SOFT**
COMPUTERSYSTEME
Händelstr. 2-4 · 5000 Köln 1
(0221) 211761
Schriftliche Händleranfragen



**ALBATROS-COMPUTER GmbH**

**7107 BAD WIMPFEN, MAGNUSSTR. 9, TEL. 0 70 63/70 57 (KEIN LADENVERKAUF)**

**→ EIN NAME, DEN MAN SICH MERKEN SOLLTE. ←**

Computer aus eigener Fertigung. Unser Service endet nicht mit der Auslieferung. Wir bauen Ihnen jeden Computer bis hin zum 20 Mhz AT mit 16 MHz Coprozessor oder auch 24 MHz 386. Auch in der Meß- und Betriebsdatenerfassung sind wir ein kompetenter Partner für Problemlösungen.

**Zur Zeit bieten wir an:**

**AC 88/03**, XT compatibel mit 360 KB FDD und 20 MB HDD, 512 KB RAM (max. 640 KB on board) 4,77/10 MHz schaltbar, Mainboard mit Monochrom/Color/Graphik Controller, Multi I/O mit parallel, seriell, Mouse, Game und Echtzeituhr, Diskettencontroller für 360 KB und 1,2 MB Laufwerke, HDD Controller, 150 Watt Netzteil, AT-Look Gehäuse mit LED's und Schalter, deutsche Tastatur, MS-DOS und 12" Monitor grün Preis komplett: **2.099,– DM**

**AC 286/03**, Baby AT-kompatibel 16 MHz Speed mit 1,2 MB FDD und 41 MB HDD, 640 KB 0 Wait (max. 4 MB on board) 8 Slots, Legal Bios, Echtzeituhr, schneller Kombicontroller für 2 FDD und 2 HDD, AT I/O Karte mit parallel und seriell Schnittstelle, monochrome Graphikkarte, 101 Tasten deutsch, 200 Watt Netzteil, Baby Gehäuse mit LED's und Schaltern, 14" Monochrome Monitor Flatscreen amber mit Drehfuß Preis komplett: **3.699,– DM**

**AC 386/01**, 80386 CPU, 16 MHz (21 MHz Speed) mit 1,2 MB FDD und 20 MB HDD, 2 MB RAM 0 Wait (max. 8 MB on board) 8 Slots, Kombicontroller für Interleave 1:1, I/O Karte mit parallel und seriell, Monochrome/Graphik Karte mit 720 x 348 Auflösung und Printerport, 101 Tasten deutsch, 200 Watt Netzteil, Baby-Gehäuse mit LED's und Schalter, 14" Monochrome Monitor Flatscreen amber mit Drehfuß Preis komplett: **6.698,– DM**

Portables mit LCD 640 x 400 XT ab 2.500,– DM, AT ab 3.400,– DM

**Ab sofort auch Software im Angebot: z.B. dBFast dB-Compiler 215,– DM**

Lieferung erfolgt ab Lager solange der Vorrat reicht. Diese Angebote werden nur im Versandhandel geliefert, Händler fordern unsere Preisliste mit Gewerbeanmeldung an.
**Wir suchen auch weiterhin noch Stützpunkthändler mit Gebietsschutz in ganz Europa.** Interessenten aus dem Ausland wollen bitte die 14 % MwSt aus unseren Angebotspreisen wieder abziehen. Die Versandkosten für ein Komplettgerät mit Monitor in einer wirklich stabilen Verpackung (Made in Germany) betragen ca. 45,– DM, dafür kommen die Rechner auch wirklich unbeschädigt an.

**ALBATROS-COMPUTER GmbH**

**7107 BAD WIMPFEN, MAGNUSSTR. 9, TEL. 0 70 63/70 57 (KEIN LADENVERKAUF)**

**→ EIN NAME, DEN MAN SICH MERKEN SOLLTE. ←**

```
                                }
[L2C]   88:   { button = 0      : linke Taste
                                }
[ADC]   89:   { button = 1      : rechte Taste
                                }
[XTG]   90:   { ButtonPressed   : button jetzt gedrueckt
      ?                         }
[TDE]   91:   { CountOFPressed  : Anzahl der clicks
                                }
[ZCE]   92:   { x,y             : Position beim letzten
      click                     }
[000]   93:
[OX4]   94:   VAR Reg : RegTyp;
[000]   95:
[B21]   96:   BEGIN
[LC3]   97:     Reg.ax:=$0005;
[D15]   98:     Reg.bx:=button;
[2Y2]   99:     Intr(51,Reg);
[ZD3]  100:     IF button = 0
[TN9]  101:       THEN ButtonPressed:=  (Reg.ax AND $01
      )<>0
[LT9]  102:       ELSE ButtonPressed:=  (Reg.ax AND $02
      )<>0;
[8P7]  103:     CountOfPressed:=Reg.bx;
[GR2]  104:     x:= Reg.cx;
[KR2]  105:     y:= Reg.dx;
[TMO]  106:   END;
[Z7C]  107: (*---------------------------------------
      ----------------------------------*)
[000]  108:
[44H]  109: PROCEDURE GetButtonReleaseInfo ( button
                : Integer;
[BG8]  110:                                   VAR Button
      Pressed   : Boolean;
[7N8]  111:                                   VAR CountO
      fReleased : Integer;
[YD6]  112:                                   VAR x,y
                : Integer );
[000]  113:
[CKL]  114:   { Curorposition beim letzten loslassen de
      s button                  }
[OYD]  115:   { button = 0      : linke Taste
                                }
[FUD]  116:   { button = 1      : rechte Taste
                                }
[08K]  117:   { ButtonPressed   : button jetzt gedrueck
      t ?                       }
[TSN]  118:   { CountOFReleased : Anzahl der clicks  (
      beim losslassen gezaehlt )  }
[44M]  119:   { x,y             : Position beim letzten
       losslassen des button    }
[000]  120:
[OX4]  121:   VAR Reg : RegTyp;
[000]  122:
[B21]  123:   BEGIN
[SC3]  124:     Reg.ax:=$0006;
[D15]  125:     Reg.bx:=button;
[2Y2]  126:     Intr(51,Reg);
[ZD3]  127:     IF button = 0
[TN9]  128:       THEN ButtonPressed:=  (Reg.ax AND $01
      )<>0
[LT9]  129:       ELSE ButtonPressed:=  (Reg.ax AND $02
      )<>0;
[F48]  130:     CountOfReleased:=Reg.bx;
[GR2]  131:     x:= Reg.cx;
[KR2]  132:     y:= Reg.dx;
[TMO]  133:   END;
[Z7C]  134: (*---------------------------------------
      ----------------------------------*)
[000]  135:
[ZFC]  136: PROCEDURE SetMouseWindow ( x, y : Integer )
      ;
[000]  137:
[HRM]  138:   { setzt die Koordinaten in denen die Maus
       bewegt werden kann }
[YWD]  139:   { x, y  maximal 639, 349  ( im TextMode 6
      39, 199 )           }
[000]  140:
[SY4]  141:   VAR Reg     : RegTyp;
[000]  142:
[B21]  143:   BEGIN
[1G2]  144:     Reg.ax:=7;
[4G2]  145:     Reg.cx:=0;
[HK2]  146:     Reg.dx:=x;
[KRC]  147:     Intr(51,Reg);     { set horizonal posit
      ion }
[2G2]  148:     Reg.ax:=8;
[4G2]  149:     Reg.cx:=0;
[KK2]  150:     Reg.dx:=y;
[A6C]  151:     Intr(51,Reg);     { set vertical positi
      on }
[TMO]  152:   END;
[Z7C]  153: (*---------------------------------------
      ----------------------------------*)
[000]  154:
[18B]  155: PROCEDURE SetGraphCursor (i : Integer);
```

```
[000]  156:
[A7D]  157:   { bestimmt die Form des Grafikcursors.
      }
[CM9]  158:   { 0  : Standard Cursor Shape
      }
[E97]  159:   { 1  : Up Arrow
      }
[E37]  160:   { 2  : Down Arrow
      }
[R27]  161:   { 3  : Left Arrow
      }
[737]  162:   { 4  : Right Arrow
      }
[HU6]  163:   { 5  : Check Mark
      }
[Y28]  164:   { 6  : Pointing Hand
      }
[BG8]  165:   { 7  : Diagonal Cross
      }
[B79]  166:   { 8  : Rectangular Cross
      }
[8E7]  167:   { 9  : Hourglass
      }
[7Z8]  168:   { 10 : Smal Standard Cursor
      }
[000]  169:
[000]  170:
[LSA]  171:   TYPE   Koordinaten = ARRAY [1..2] OF Inte
      ger;
[000]  172:
[HB5]  173:          CursorShape = RECORD
[DK9]  174:                          Mask    : ARRAY [
      0..1,0..15] OF Integer;
[48H]  175:                          HotSpot : Koordin
      aten;  { dieser Punkt bestimmt den }
[478]  176:
                { x, y Wert bei einem Aufruf}
[TMO]  177:                          END;
[000]  178:
[P6G]  179:   CONST  Mask00 : CursorShape =        { st
      andard cursor shape }
[1AC]  180:          ( Mask : ( ( $3FFF, $1FFF, $0FFF,
      $07FF, $03FF, $01FF, $00FF, $007F,
[1G9]  181:                     $003F, $001F, $01FF,
      $10FF, $30FF, $F87F, $F87F, $FC3F ),
[4T8]  182:                   ( $0000, $4000, $6000,
      $7000, $7800, $7C00, $7E00, $7F00,
[NV8]  183:                     $7F80, $78C0, $7C00,
      $4600, $0600, $0300, $0300, $0180 ) );
[08A]  184:               HotSpot : (0,0)
                                              );
[000]  185:
[U3D]  186:          Mask01 : CursorShape =        { up
       arrow                }
[Z2C]  187:          ( Mask : ( ( $F9FF, $F0FF, $E07F,
      $E07F, $C03F, $C03F, $801F, $801F,
[MV9]  188:                     $000F, $000F, $F0FF,
      $F0FF, $F0FF, $F0FF, $F0FF, $F0FF ),
[7Z8]  189:                   ( $0000, $0600, $0F00,
      $0F00, $1F80, $1F80, $3FC0, $3FC0,
[HT8]  190:                     $7FE0, $0600, $0600,
      $0600, $0600, $0600, $0600, $0000 ) );
[SGA]  191:               HotSpot : (5,0)
                                              );
[000]  192:
[S5D]  193:          Mask02 : CursorShape =        { do
      wn arrow             }
[SEC]  194:          ( Mask : ( ( $F0FF, $F0FF, $F0FF,
      $F0FF, $F0FF, $F0FF, $000F, $000F,
[NR9]  195:                     $801F, $801F, $C03F,
      $C03F, $E07F, $E07F, $F0FF, $F9FF ),
[8S8]  196:                   ( $0000, $0600, $0600,
      $0600, $0600, $0600, $0600, $7FE0,
[W39]  197:                     $3FC0, $3FC0, $1F80,
      $1F80, $0F00, $0F00, $0600, $0000 ) );
[EBA]  198:               HotSpot : (5,15)
                                              );
[000]  199:
[R4D]  200:          Mask03 : CursorShape =        { le
      ft arrow             }
[RKB]  201:          ( Mask : ( ( $FEFF, $F01F, $0000,
      $0000, $0000, $F01F, $FE1F, $FFFF,
[SNA]  202:                     $FFFF, $FFFF, $FFFF,
      $FFFF, $FFFF, $FFFF, $FFFF, $FFFF ),
[HT8]  203:                   ( $0000, $00C0, $07C0,
      $7FFE, $07C0, $07C0, $0000, $0000,
[FG8]  204:                     $0000, $0000, $0000,
      $0000, $0000, $0000, $0000, $0000 ) );
[PGA]  205:               HotSpot : (0,3)
                                              );
[000]  206:
[X8D]  207:          Mask04 : CursorShape =        { ri
      ght arrow            }
[CMB]  208:          ( Mask : ( ( $F87F, $F80F, $0000,
      $0000, $0000, $F80F, $F87F, $FFFF,
[SNA]  209:                     $FFFF, $FFFF, $FFFF,
```

```
          $FFFF, $FFFF, $FFFF, $FFFF, $FFFF ),
[3P8]   210:                        ( $0000, $0200, $03E0,
          $7FFE, $03E0, $0200, $0000, $0000,
[FG8]   211:                          $0000, $0000, $0000,
          $0000, $0000, $0000, $0000, $0000 ) );
[4BA]   212:              HotSpot : (16,3)
                                                        );
[000]   213:
[KOD]   214:           Mask05 : CursorShape =         { ch
          eck mark                 }
[37C]   215:           ( Mask : ( ( $FFF0, $FFE0, $FFC0,
          $FF81, $FF03, $0607, $000F, $001F,
[BEA]   216:                          $C03F, $F07F, $FFFF,
          $FFFF, $FFFF, $FFFF, $FFFF, $FFFF ),
[TR8]   217:                        ( $0000, $0006, $000C,
          $0018, $0030, $0060, $70C0, $1D80,
[4H8]   218:                          $0700, $0000, $0000,
          $0000, $0000, $0000, $0000, $0000 ) );
[6KA]   219:              HotSpot : (6,7)
                                                        );
[000]   220:
[CAE]   221:           Mask06 : CursorShape =         { po
          inting hand              }
[92C]   222:           ( Mask : ( ( $E1FF, $E1FF, $E1FF,
          $E1FF, $E1FF, $0000, $0000, $0000,
[S88]   223:                          $0000, $0000, $0000,
          $0000, $0000, $0000, $0000, $0000 ),
[2M8]   224:                        ( $1E00, $1200, $1200,
          $1200, $1200, $13FF, $1249, $1249,
[V69]   225:                          $F249, $9001, $9001,
          $9001, $8001, $8001, $8001, $FFFF ) );
[SGA]   226:              HotSpot : (5,0)
                                                        );
[000]   227:
[USE]   228:           Mask07 : CursorShape =         { di
          agonal cross             }
[D5B]   229:           ( Mask : ( ( $07E0, $0180, $0000,
          $C003, $F00F, $C003, $0000, $0180,
[DFA]   230:                          $07E0, $FFFF, $FFFF,
          $FFFF, $FFFF, $FFFF, $FFFF, $FFFF ),
[DV8]   231:                        ( $0000, $700E, $1C38,
          $0660, $03C0, $0660, $1C38, $700E,
[FG8]   232:                          $0000, $0000, $0000,
          $0000, $0000, $0000, $0000, $0000 ) );
[SHA]   233:              HotSpot : (7,4)
                                                        );
[000]   234:
[P3F]   235:           Mask08 : CursorShape =         { re
          ctangular cross          }
[TRB]   236:           ( Mask : ( ( $FC3F, $FC3F, $FC3F,
          $0000, $0000, $0000, $FC3F, $FC3F,
[3LA]   237:                          $FC3F, $FFFF, $FFFF,
          $FFFF, $FFFF, $FFFF, $FFFF, $FFFF ),
[ZM8]   238:                        ( $0000, $0180, $0180,
          $0180, $7FFE, $0180, $0180, $0180,
[FG8]   239:                          $0000, $0000, $0000,
          $0000, $0000, $0000, $0000, $0000 ) );
[SHA]   240:              HotSpot : (7,4)
                                                        );
[000]   241:
[MBD]   242:           Mask09 : CursorShape =         { ho
          urglass                  }
[AZA]   243:           ( Mask : ( ( $0000, $0000, $0000,
          $0000, $8001, $C003, $E007, $F00F,
[729]   244:                          $E007, $C003, $8001,
          $0000, $FFFF, $0000, $0000, $FFFF ),
[019]   245:                        ( $0000, $7FFE, $6006,
          $300C, $1818, $0C30, $0660, $03C0,
[549]   246:                          $0660, $0C30, $1998,
          $33CC, $67E6, $7FFE, $0000, $0000 ) );
[6HA]   247:              HotSpot : (7,1)
                                                        );
[000]   248:
[LGG]   249:           Mask10 : CursorShape =         { sm
          al standard cursor shape }
[46D]   250:           ( Mask : ( ( $FFFF, $FFFF, $FFFF,
          $FFFF, $FFFF, $FFFF, $FFFF, $FFFF,
[SNA]   251:                          $FFFF, $FFFF, $FFFF,
          $FFFF, $FFFF, $FFFF, $FFFF, $FFFF ),
[FM8]   252:                        ( $8000, $E000, $F800,
          $FE00, $D800, $0C00, $0600, $0300,
[FG8]   253:                          $0000, $0000, $0000,
          $0000, $0000, $0000, $0000, $0000 ) );
[A8A]   254:              HotSpot : (1,1)
                                                        );
[000]   255:
[000]   256:
[OT5]   257:    VAR Reg          : RegTyp;
[6DE]   258:        Cursor       : ARRAY [1..11] OF Cursor
          Shape ABSOLUTE Mask01;
[000]   259:
[B21]   260:    BEGIN
[000]   261:
[3G2]   262:      Reg.ax:=9;
[BA9]   263:      Reg.bx:=Cursor[i].HotSpot [1];
[MA9]   264:      Reg.cx:=Cursor[i].HotSpot [2];
```


**PLOTTER PL 22 B**

**ein Plotter zum Selberbauen**

- Centronicsschnittstelle mit 8 KB Puffer
- Auflösung 0,03
- Genauigkeit besser 0.1
- stabile Metallkonstruktion
- Antrieb durch Zahnriemen, dadurch schlupffrei
- Zeichenfläche 395 x 280 DIN A 3
- serielle Schnittstelle mit HP-GL, dadurch entfallen Anpassungsprobleme
- und 10 KB Puffer

**Mechanikbausatz DM 619,95**

Hardwarepaket
„Rechner-Plotter mit Text-CAD"
– keine Anpassungsprobleme mehr –
**DM 2998,50**

**INFO** gegen DM 1,60 in Briefmarken bei:

**PROFAST**
**Plotter- und Grafiksysteme**
Buchbergstraße 37
D-7712 Blumberg



FONTASY®

**Das Schriftarten- und Zeichenprogramm für IBMPC/XT/AT und Kompatible mit über 400 verschiedenen Schriften, Rändern, Symbolen und Zeichnungen.**

**In der Grundausstattung sind bereits 28 Schriften enthalten.**

**Viele der angebotenen Fonts wurden von uns um die deutschen Umlaute ergänzt.**

**Jetzt ist auch ein deutsches Handbuch verfügbar.**

**FONTASY - Version 2 ...DM 168,00**
**zusätzliche**
**Font-Disketten .....ab 49,90**
**Deutsches Handbuch ...DM 39,00**

**Individuelle Daten Technik**
**Osterfeldstr. 17**
**8901 Stadtbergen,**
**Tel.: (0821) 43 52 71**



**Wer — wo — wann?**

Bei Zuschriften bitten wir Folgendes zu beachten: Haben Sie Fragen, Anregungen oder Programme, die Sie der DOS-Redaktion zukommen lassen möchten, adressieren Sie wie folgt:

**Redaktion**
**DOS International**
**Hans-Stießberger-Str. 3**
**8013 Haar/München**
**Tel. (089) 46 10 01-0**

Haben Sie Fragen, die zum Beispiel Abonnements, Kleinanzeigen oder Databox-Bestellungen betreffen, wenden Sie sich bitte an den Verlag:

**DMV Verlag GmbH**
**DOS International**
**Fuldaer Straße 6**
**3440 Eschwege**
**Tel. (0 56 51) 80 09-0**
**Telex 993 210 dmv d**
**Telefax (0 56 51) 80 09-33**

```
[E07]  265:      Reg.dx:=Ofs(Cursor[i]);
[5H3]  266:      Reg.es:=CSeg;
[2Y2]  267:      Intr(51,Reg);
[TMO]  268:    END;
[Z7C]  269: (*-----------------------------------------
       ----------------------------------*)
[000]  270:
[K9B]  271: PROCEDURE SetTextCursor ( i : Integer);
[000]  272:
[UND]  273:    { Setzen eines Mauscursors im Textmodus
       }
[04B]  274:    { i : 0-3 welcher Softwarecursor
       }
[GTD]  275:    { Musterbestimmung siehe MS-Mouse Manual
       }
[000]  276:
[RUB]  277:    CONST TextCursor : ARRAY[0..3,1..2] OF In
       teger
[7HA]  278:                      = ( ($FFFF, $7F00), ($
       0000, $0D04), ($0000, $3D01), ($0000, $4F0F)
       );
[000]  279:
[4W4]  280:    VAR   Reg        : RegTyp;
[000]  281:
[B21]  282:    BEGIN
[PM2]  283:      Reg.ax:=10;
[ZF2]  284:      Reg.bx:=0;
[MB7]  285:      Reg.cx:=TextCursor[i,1];
[ZB7]  286:      Reg.dx:=TextCursor[i,2];
[2Y2]  287:      Intr(51,Reg);
[TMO]  288:    END;
[Z7C]  289: (*-----------------------------------------
       ----------------------------------*)
[000]  290:
[ZAF]  291: PROCEDURE ReadMouseMotionCounter ( VAR x, y
       : Integer );
[000]  292:
[OHR]  293:    { liefert die Bewegung der Maus seit dem
       letzten Aufruf dieser Funktion.
[0P5]  294:      Angaben in 1/200 inch
[1DA]  295:      x positiv : Bewegung nach rechts
[5ZE]  296:      y positiv : Bewegung nach unten
                                          }
[000]  297:
[SY4]  298:    VAR Reg      : RegTyp;
[000]  299:
[B21]  300:    BEGIN
[SM2]  301:      Reg.ax:=11;
[2Y2]  302:      Intr(51,Reg);
[YB2]  303:      x:=Reg.cx;
[ZB2]  304:      y:=Reg.dx;
[TMO]  305:    END;
[Z7C]  306: (*-----------------------------------------
       ----------------------------------*)
[000]  307:
[WCD]  308: PROCEDURE SetMickeyPixelRatio ( x, y : Inte
       ger );
[000]  309:
[LSC]  310:    { Setzt die Empfindlichkeit der Maus }
[EH8]  311:    { Standard : 1 / 100   inch / pixel  }
[000]  312:
[SY4]  313:    VAR Reg      : RegTyp;
[000]  314:
[B21]  315:    BEGIN
[ON2]  316:      Reg.ax:=15;
[CK2]  317:      Reg.cx:=x;
[KK2]  318:      Reg.dx:=y;
[2Y2]  319:      Intr(51,Reg);
[TMO]  320:    END;
[Z7C]  321: (*-----------------------------------------
       ----------------------------------*)
[000]  322:
[ZSH]  323: PROCEDURE SetDoubleSpeedThreshold ( thresho
       ld : Integer );
[000]  324:
[SY4]  325:    VAR Reg      : RegTyp;
[000]  326:
[B21]  327:    BEGIN
[8N2]  328:      Reg.ax:=19;
[3H5]  329:      Reg.dx:=threshold;
[2Y2]  330:      Intr(51,Reg);
[TMO]  331:    END;
[Z7C]  332: (*-----------------------------------------
       ----------------------------------*)
```

**Listing 3. Mit »mouse.inc« haben Sie mehrere Grafikcursor zur Verfügung**

```
[FKA]    0: (*****************************************
         ******************************)
[B48]    1: (*                                       *)
[FL9]    2: (*                        D I R E K T D E . P
         A S                                        *)
[B48]    3: (*                                       *)
[F9G]    4: (*  Autor           :   Torsten Knappe Jahns
         tr. 4   4790 Paderborn                     *)
[B48]    5: (*                                       *)
[42A]    6: (*  Start           :   18.09.87
                                                    *)
[2GB]    7: (*  akt. Stand      :   21.09.87
                                                    *)
[B48]    8: (*                                       *)
[MVL]    9: (*  Prgm. Beschr.   :  Demonstriert das dire
         kte Beschreiben der EGA -                  *)
[37G]   10: (*                      Karte im hochaufloese
         nden Grafikmodus.                          *)
[B48]   11: (*                                       *)
[5UD]   12: (* Erstellt in      :   TURBO-PASCAl  V. 3.0
         1 PC-DOS                                   *)
[1DE]   13: (* getestet unter  :   PC-DOS 3.2  und PC-D
         OS 3.3                                     *)
[B48]   14: (*                                       *)
[H5F]   15: (* Rechner         :  AT 03    mit Paradise
          Autoswitch EGA                            *)
[YVB]   16: (*                                  und Colo
         rmonitor                                   *)
[46F]   17: (*                      AT Clone mit Paradise
          AutoSwitch EGA 480 und                    *)
[8UB]   18: (*                                  Colormon
         itor                                       *)
[B48]   19: (*                                       *)
[FKA]   20: (*****************************************
         ******************************)
[000]   21:
[YS4]   22: PROGRAM DirektDe;
[000]   23:
[WWB]   24:    {-----------------------------------------
        ----------------------------------}
[SXD]   25:    {$Ityp.ega}
        { Typvereinbarungen }
[WWB]   26:    {-----------------------------------------
        ----------------------------------}
[000]   27:
[2RF]   28:    VAR Font                : FontTyp;       {
        globale Variable                              }
[DHL]   29:        Akt_Speicherebenen : Integer;       {
        globale Variable, nicht direkt    }
[ON6]   30:                                            {
        benutzen !!                                   }
[W15]   31:        s1                  : StrgTyp;
[RB4]   32:        c1                  : ChTyp  ;
[UW4]   33:        i                   : Integer;
[000]   34:
[WWB]   35:    {-----------------------------------------
        ----------------------------------}
[2HH]   36:    {$Iega.inc}
        { Schreibroutinen fuer EGA-Karte  }
[WWB]   37:    {-----------------------------------------
        ----------------------------------}
[000]   38:
[H97]   39:    PROCEDURE WriteMessage;
[000]   40:
[YM4]   41:      VAR s1    : StrgTyp;
[SC4]   42:          save  : FontTyp;
[000]   43:
[B21]   44:      BEGIN
[CN2]   45:        save:=Font;
[1C5]   46:        Font.art:=Font9x14;
[UF6]   47:        GetFontPointer(Font);
[7FB]   48:        s1.x:=10; s1.y:=24; s1.inv:=TRUE; s1.
        color:=11;
[09H]   49:        s1.ch:=' Weiter durch Druecken einer
        beliebigen Taste ....';
[VR4]   50:        WriteString(s1);
[BN6]   51:        REPEAT UNTIL KeyPressed;
[YX3]   52:        DelLine(24, 11);
[4S2]   53:        Font:=Save;
[TMO]   54:      END;
[WWB]   55:    {-----------------------------------------
        ----------------------------------}
[000]   56:
[WU4]   57:    PROCEDURE Sinus;
[000]   58:
[224]   59:      VAR p1    : PTyp;
[8P3]   60:          i     : Integer;
[000]   61:
[B21]   62:      BEGIN
```

```
[CB9]   63:
                { zeichne Koordinatenkreuz }
[178]   64:       p1.color:=10; p1.funktion:=0;
[WS1]   65:       p1.x:=0;
[Z45]   66:       FOR i:=70 TO 270 DO BEGIN
[F52]   67:         p1.y:=i;
[F72]   68:         PSet(p1);
[TMO]   69:       END;
[DY1]   70:       p1.y:=170;
[Y55]   71:       FOR i:=0 TO 639 DO BEGIN
[B52]   72:         p1.x:=i;
[F72]   73:         PSet(p1);
[TMO]   74:       END;
[T99]   75:
                { zeichne Sinusschwingung  }
[UE3]   76:       p1.color:=12;
[Y55]   77:       FOR i:=0 TO 639 DO BEGIN
[B52]   78:         p1.x:=i;
[XFA]   79:         p1.y:=170 - Trunc( Sin(pi / 320 * p
        1.x) * 100);
[F92]   80:         Pset(p1);
[TMO]   81:       END;
[TMO]   82:     END;
[WWB]   83:   {-----------------------------------------
        ----------------------------------}
[000]   84:
[B21]   85:   BEGIN
[KGP]   86:     SetVideoMode(16);     { Selektiert hoch
        aufloesenden EGA - Farbgrafikmodus.}
[28H]   87:                           { Falls ein Monoc
        hromemonitor angeschlossen ist.    }
[A8E]   88:                           { muss SetVideoMo
        de(15) benutzt werden.             }
[000]   89:
[CAM]   90:     Font.art:=Font8x8;                  { kl
        eine Schrift soll angezeigt werden }
[3KL]   91:     GetFontpointer(Font);
          { Initialisierung der 8x8 Schrift }
[FKN]   92:     SelectSpeicherEbene(3210);  { alle Eben
        en zum Schreiben freigegeben, d.h. }
[5KB]   93:                                 { alle 16 F
        arben sind erlaubt.                }
[OVG]   94:     ShowBildEbene(3210);
           { zeige alle Ebenen (Farben) an }
[000]   95:
[ZHB]   96:     s1.x:=0; s1.y:=0; s1.inv:=FALSE; s1.col
        or:=15;
[TRL]   97:     s1.ch:='Dies ist eine Stringausgabe, di
        e bei Position 0, 0 beginnt.';
[VR4]   98:     WriteString(s1);
[AT1]   99:     s1.y:=1;
[2MN]  100:     s1.ch:='Es wird der 8x8 Font der EGA-Ka
        rte verwendet. Alle Textausgaben';
[VR4]  101:     WriteString(s1);
[HT1]  102:     s1.y:=2;
[H3B]  103:     s1.ch:='erfolgen im Grafikmodus !!!!';
[VR4]  104:     WriteString(s1);
[NZ3]  105:     WriteMessage;
[000]  106:
[1C5]  107:     Font.art:=Font9x14;
[UF6]  108:     GetFontpointer(Font);
[ST1]  109:     s1.y:=3;
[WPR]  110:     s1.ch:='Dadurch ist es auch möglich zwi
        schendurch mal die Schriftgroesse';
[VR4]  111:     WriteString(s1);
[ZT1]  112:     s1.y:=4;
[SAP]  113:     s1.ch:='zu wechseln. Beachten Sie bitte
         , dass die Anzahl der moeglichen';
[VR4]  114:     WriteString(s1);
[6U1]  115:     s1.y:=5;
[8HA]  116:     s1.ch:='Zeilen unterschiedlich ist.';
[VR4]  117:     WriteString(s1);
[NZ3]  118:     WriteMessage;
[000]  119:
[Y65]  120:     Font.art:=Font8x8;
[UF6]  121:     GetFontpointer(Font);
[3P5]  122:     s1.y:=25; s1.color:=13;
[HCP]  123:     s1.ch:='Natuerlich koennen auch die Far
        ben veraendert werden. Mithilfe';
[VR4]  124:     WriteString(s1);
[CV1]  125:     s1.y:=26;
[EGP]  126:     s1.ch:='von StrgTyp.color, SelectSpeich
        erEbene() und ShowSpeicherEbene()';
[VR4]  127:     WriteString(s1);
[CV1]  128:     s1.y:=27;
[9WP]  129:     s1.ch:='koennen vielfaeltige Varianten
        der Farbgebung angewandt werden.';
[VR4]  130:     WriteString(s1);
[NZ3]  131:     WriteMessage;
[NF2]  132:     ClrScr;
[000]  133:
[1C5]  134:     Font.art:=Font9x14;
[UF6]  135:     GetFontpointer(Font);
[3T1]  136:     s1.y:=0;
[NEP]  137:     s1.ch:='Als naechstes werden einzelne Z
        eichen ausgegeben. Die verwendete';
```


R&P GMBH
BOXGRABEN 131-133
D-5100 AACHEN
TEL. (02 41) 40 24 07
+ 40 24 71
FAX (02 41) 40 24 20

dBü

WIR WÜRDEN UNSER DBÜ GERN MIT ANDEREN PRODUKTEN VERGLEICHEN ...
FRAGT SICH NUR WOMIT ? ? !

E S G I B T
K E I N E N
A N D E R E N

dBASE*-COMPILER/
dBASE* NACH C-ÜBERSETZER
MIT VERGLEICHBAREN
LEISTUNGSMERKMALEN

DOCH
URTEILEN
SIE SELBST

dBü

- C-QUELLCODE AUS dBASE*-CODE
- LAUFFÄHIGE EXE.-FILES
- LIZENZFREIE WEITERGABE DER MIT DBÜ ERSTELLTEN EXE.-FILES
- HOHE PORTABILITÄT DER ERSTELLTEN dBASE*-PROGRAMME
- UMFANGREICHER BEFEHLSVORRAT PROMPT/MENU/SAVE/RESTORE SCREEN
- NETZWERKFÄHIGES PROGRAMMERGEBNIS
- INTEGRIERTER C COMPILER/LINKER
- DEUTSCHE BENUTZEROBERFLÄCHE
- DEUTSCHES HANDBUCH

GEBEN SIE SICH NICHT MIT WENIGER ZUFRIEDEN!

BRINGEN AUCH SIE ÜBERRASCHENDES TEMPO IN IHRE dBASE*-PROGRAMME! AB SOFORT ERSTELLEN SIE C PROGRAMME, AUCH WENN SIE DIESE PROGRAMMIER-SPRACHE NICHT BEHERRSCHEN.

dBü DM 498,– dBü JUNIOR DM 348,–

dBü DEMODISKETTE/DATENBLÄTTER DM 10,–

EINFÜHRUNGSPREIS (BIS 30. 09. '88)

WIR SUCHEN NOCH EXKLUSIV-DISTRIBUTOREN EUROPAWEIT.

*dBASE IST EIN EINGETRAGENES WARENZEICHEN DER FIRMA ASHTON-TATE

```
[VR4]  138:      WriteString(s1);
[AT1]  139:      s1.y:=1;
[S8P]  140:      s1.ch:='Prozedur WriteChar arbeitet vol
         lkommen analog zu WriteString,  ';
[VR4]  141:      WriteString(s1);
[HT1]  142:      s1.y:=2;
[PMH]  143:      s1.ch:='nur das hier einzelne Zeichen a
         usgegeben werden.';
[VR4]  144:      WriteString(s1);
[NZ3]  145:      WriteMessage;
[WC6]  146:      c1.y:=10; c1.inv:=True;
[U25]  147:      FOR i:=0 TO 15 DO BEGIN
[NS6]  148:        c1.x:=(i * 3) + 10; c1.color:=i;
[LT5]  149:        c1.ch:=Chr(Ord('A') + i );
[7W3]  150:        WriteChar(c1);
[TMO]  151:      END;
[NZ3]  152:      WriteMessage;
[000]  153:
[Y65]  154:      Font.art:=Font8x8;
[UF6]  155:      GetFontpointer(Font);
[2Y5]  156:      c1.y:=20; c1.inv:=FALSE;
[U25]  157:      FOR i:=0 TO 15 DO BEGIN
[NR8]  158:        c1.x:=(i * 3); c1.color:=(i + 5) Mod
         16;
[LT5]  159:        c1.ch:=Chr(Ord('A') + i );
[7W3]  160:        WriteChar(c1);
[TMO]  161:      END;
[NZ3]  162:      WriteMessage;
[NF2]  163:      ClrScr;
[000]  164:
[1C5]  165:      Font.art:=Font9x14;
[UF6]  166:      GetFontpointer(Font);
[BV8]  167:      s1.y:=0; s1.color:=11; s1.inv:=FALSE;
[08G]  168:      s1.ch:='Zum Abschluss wird die Punktaus
         gabe gezeigt.';
[VR4]  169:      WriteString(s1);
[AT1]  170:      s1.y:=1;
[C7H]  171:      s1.ch:='Hierzu wird eine Sinusschwingun
         g ausgegeben.';
[VR4]  172:      WriteString(s1);
[ST1]  173:      s1.y:=3;
[8YH]  174:      s1.ch:='Vorher wird noch ein Koordinate
         nkreuz gezeichnet.';
[VR4]  175:      WriteString(s1);
[NZ3]  176:      WriteMessage;
[4Z1]  177:      Sinus;
[NZ3]  178:      WriteMessage;
[063]  179:      InvertBild;
[3P5]  180:      s1.y:=21; s1.color:=13;
[5KN]  181:      s1.ch:='Mit dem Invertieren des Bildes
         ist das Demoprogramm beendet.';
[VR4]  182:      WriteString(s1);
[CV1]  183:      s1.y:=22;
[EVF]  184:      s1.ch:='Viel Spaß beim benutzen der Pro
         zeduren !!!!!!!!!';
[VR4]  185:      WriteString(s1);
[NZ3]  186:      WriteMessage;
[000]  187:
[R6K]  188:      SetVideoMode(3);          { zurueck zum
         'normalen' Textmodus }
[5LO]  189:    END.
```

**Listing 4. »direktde.pas« demonstriert den Aufruf der Routinen**

```
[FKA]    0: (*****************************************
           ******************************)
[B48]    1: (*
                                          *)
[1S9]    2: (*                    M A U S D E M O . P
           A S                            *)
[B48]    3: (*
                                          *)
[F9G]    4: (*  Autor          :   Torsten Knappe Jahns
           tr. 4   4790 Paderborn         *)
[B48]    5: (*
                                          *)
[W1A]    6: (*  Start          :   10.09.87
                                          *)
[2GB]    7: (*  akt. Stand     :   21.09.87
                                          *)
[B48]    8: (*
                                          *)
[KCM]    9: (*  Prgm. Beschr.  :  Demonstriert den Gebr
           auch einer Maus mit einer      *)
[HPH]   10: (*                   EGA-Karte im hochaufl
           oesenden Grafikmodus.          *)
[B48]   11: (*
                                          *)
```

```
[5UD]   12: (* Erstellt in      :   TURBO-PASCAl  V. 3.0
           1 PC-DOS                       *)
[1DE]   13: (* getestet unter  :   PC-DOS 3.2  und PC-D
           OS 3.3                         *)
[B48]   14: (*
                                          *)
[39G]   15: (* Rechner         :  AT 03    mit Paradise
           Autoswitch EGA und             *)
[OTF]   16: (*                                  Colormon
           itor  sowie einer MS-Mouse     *)
[46F]   17: (*                    AT Clone mit Paradise
           AutoSwitch EGA 480 und         *)
[DUG]   18: (*                                  Colormon
           itor sowie einer Genius-Maus   *)
[B48]   19: (*
                                          *)
[FKA]   20: (*****************************************
           ******************************)
[000]   21:
[UU4]   22: PROGRAM MausDemo;
[000]   23:
[WWB]   24:   {--------------------------------------
           ----------------------------------}
[SXD]   25:   {$Ityp.ega}
           { Typvereinbarungen }
[WWB]   26:   {--------------------------------------
           ----------------------------------}
[000]   27:
[WR8]   28:   VAR x, y, AltX, AltY, i   : Integer;
[BYF]   29:       lp, rp                : Boolean;    {
           Left (Right) Key Pressed ?          }
[P9E]   30:       Font                  : FontTyp;    {
           globale Variable                    }
[VNL]   31:       Akt_Speicherebenen    : Integer;    {
           globale Variable, nicht direkt      }
[ON6]   32:                                           {
           benutzen !!                         }
[PG6]   33:       sx, sy                : String[5];
[R16]   34:       s1, s2                : StrgTyp;
[000]   35:
[000]   36:
[WWB]   37:   {--------------------------------------
           ----------------------------------}
[2HH]   38:   {$Iega.inc}
           { Schreibroutinen fuer EGA-Karte  }
[CEF]   39:   {$Imouse.inc}
           { Mausroutinen                    }
[WWB]   40:   {--------------------------------------
           ----------------------------------}
[000]   41:
[H97]   42:   PROCEDURE WriteMessage;
[000]   43:
[YM4]   44:     VAR s1   : StrgTyp;
[SC4]   45:         save : FontTyp;
[000]   46:
[B21]   47:     BEGIN
[CN2]   48:       save:=Font;
[1C5]   49:       Font.art:=Font9x14;
[UF6]   50:       GetFontPointer(Font);
[7FB]   51:       s1.x:=10; s1.y:=24; s1.inv:=TRUE; s1.
           color:=11;
[09H]   52:       s1.ch:=' Weiter durch Druecken einer
           beliebigen Taste ....';
[VR4]   53:       WriteString(s1);
[BN6]   54:       REPEAT UNTIL KeyPressed;
[YX3]   55:       DelLine(24, 11);
[4S2]   56:       Font:=Save;
[TMO]   57:     END;
[WWB]   58:   {--------------------------------------
           ----------------------------------}
[000]   59:
[B21]   60:   BEGIN
[KGP]   61:     SetVideoMode(16);     { Selektiert hoch
           aufloesenden EGA - Farbgrafikmodus.}
[28H]   62:                           { Falls ein Monoc
           hrememonitor angeschlossen ist,     }
[A8E]   63:                           { muss SetVideoMo
           de(15) benutzt werden.              }
[000]   64:
[CAM]   65:     Font.art:=Font8x8;                    { kl
           eine Schrift soll angezeigt werden }
[3KL]   66:     GetFontpointer(Font);
          { Initialisierung der 8x8 Schrift }
[000]   67:
[FKN]   68:     SelectSpeicherEbene(3210);  { alle Eben
           en zum Schreiben freigegeben, d.h. }
[5KB]   69:                                   { alle 16 F
           arben sind erlaubt.                 }
[OVG]   70:     ShowBildEbene(3210);
          { zeige alle Ebenen (Farben) an }
[000]   71:
[GHB]   72:     s1.x:=0; s1.y:=0; s1.inv:=FALSE; s1.col
           or:=12;
[VFR]   73:     s1.ch:='Dieses Programm demonstriert di
           verse Mauscursor im Grafikmodus.';
[VR4]   74:     WriteString(s1);
[AT1]   75:     s1.y:=1;
[W8N]   76:     s1.ch:='Alle Mauskoordinaten werden in
           Pixelkoordinaten angegeben !';
```

```
[VR4]   77:      WriteString(s1);
[HT1]   78:      s1.y:=2;
[X3P]   79:      s1.ch:='Druecken der linken Taste schal
           tet den naechsten Cursor ein. ';
[VR4]   80:      WriteString(s1);
[ST1]   81:      s1.y:=3;
[UPH]   82:      s1.ch:='druecken der rechten Taste been
           det das Programm. ';
[VR4]   83:      WriteString(s1);
[NZ3]   84:      WriteMessage;
[OOO]   85:
[M1G]   86:      IF NOT  MouseInstalled
                           { Maus vorhanden ?  }
[ZY1]   87:        THEN BEGIN
[1C5]   88:               Font.art:=Font9x14;
[UF6]   89:               GetFontpointer(Font);
[PFB]   90:               s1.x:=10; s1.y:=10; s1.inv:=TR
           UE; s1.color:=15;
[9N9]   91:               s1.ch:='                    A C
           H T U N G !!!!!!!                  ';
[VR4]   92:               WriteString(s1);
[5V1]   93:               s1.y:=11;
[2OM]   94:               s1.ch:=' Es ist kein Microsoft
            kompatibler Maustreiber geladen ! ';
[VR4]   95:               WriteString(s1);
[5V1]   96:               s1.y:=12;
[CXD]   97:               s1.ch:=' Laden Sie bitte den T
           reiber.                            ';
[VR4]   98:               WriteString(s1);
[NZ3]   99:               WriteMessage;
[DDO]  100:             END
[YY1]  101:      ELSE BEGIN
[ETD]  102:               MouseCursorOn;
                          { MausCursor an       }
[NCF]  103:               SetMouseWindow(639,349);
                          { maximales Window    }
[STF]  104:               SetGraphCursor(1);
                          { starte mit Cursor 1 }
[4S7]  105:               s1.x:=1; s1.y:=33; s1.inv:=TRU
           E;
[R63]  106:               s1.color:=0;
[1Z3]  107:               s1.ch:='                    ';
```

```
[VV1]  108:               s2:=s1;
[S74]  109:               s2.inv:=False;
[BF3]  110:               s2.color:=15;
[A34]  111:               AltX:=0; AltY:=0;
[XG1]  112:               REPEAT
[92F]  113:                 GetMouseInfo(x, y, lp, rp);
                              { lese Mauswerte }
[OOO]  114:
[PM9]  115:                 IF ( NOT ( AltX IN [ (x - 5)
           .. ( x + 5) ]) ) OR
[6BA]  116:                      ( NOT ( ALTY IN [ (y - 5)
           .. ( y + 5) ]) )    THEN BEGIN
[VR4]  117:                   WriteString(s1);
[3W3]  118:                   Str(x : 5, sx);
[DW3]  119:                   Str(y : 5, sy);
[734]  120:                   s2.ch:=sx + sy;
[DFD]  121:                   WriteString(s2);
                              { aktuelle Werte }
[NO6]  122:                   Delay(50);

[TMO]  123:                 END;
[CV3]  124:                 IF lp THEN BEGIN
[WW2]  125:                   i:=Succ(i);
[ZCF]  126:                   SetGraphCursor(i Mod 10);
                            { naechsten Cursor }
[CKA]  127:                   Delay(500);
                            { warte ...        }
[TMO]  128:                 END;
[M4G]  129:               UNTIL rp;                    {
           Druecken der rechten Taste beendet }
[956]  130:                                            {
           das Demo.                                  }
[TMO]  131:             END;
[1EF]  132:      MouseCursorOff;
                          { MausCursor aus      }
[GTF]  133:      SetVideoMode(3);
                          { normalen Textmodus }
[5LO]  134:   END.
```

**Listing 5. »mousedemo.pas« zeigt Ihnen den Gebrauch der Maus mit der EGA-Karte**


**Software PC 51/4":**

**STARDIVISION**

| | |
|---|---|
| Star-Writer PC | 398,-- |
| Star-Planer PC | 298,-- |
| Business-Star PC | 910,-- |

**GFA-Systemtechnik**

| | |
|---|---|
| GFA-DRAFT plus | 348,-- |
| GFA-FAKT | 149,-- |
| GFA-DESK | 99,-- |

**Sybex-Software**

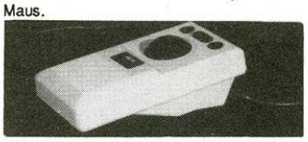
| | |
|---|---|
| Textverarbeitung | 125,-- |
| Adreßverwaltung | 95,-- |
| Dateiverwaltung | 145,-- |
| Fakturierung | 175,-- |
| Finanzbuchhaltung | 350,-- |
| Artikel und Lager | 175,-- |
| Lohn und Gehalt | 175,-- |
| DOS-Manager | 75,-- |

**DATA BECKER**

| | |
|---|---|
| DATAMAT | 99,-- |
| TEXTOMAT | 99,-- |
| BECKERstart | 298,-- |
| BECKERbase | 99,-- |

**Marconi RB2 Trackerball**

Der MARCONI Trackerball eignet sich hervorragend im CAD/CAM-Bereich und der Textverarbeitung u.v.m.
Dank der kompakten Bauweise benötigen Sie **keinen** Platz mehr zum Bewegen der Maus.



**Händleranfragen erwünscht !**
**Einzelinfo schriftlich anfordern !**

| | |
|---|---|
| Schneider/AMSTRAD PC | 198,-- |
| IBM-Kompatible (benötigen RS232) | 498,-- |
| Atari ST / Amiga | 198,-- |

**Festplatten • Drivecards**

• **20 MB** Festplatte
für Schneider/AMSTRAD 1512/1640 mit Controller, Lüfter und Kabel in Stahlgehäuse. Anschlußfertig! (Problemlos selbst einbaubar). **DM 848,--**

• **Tandon** Drivecard 20 MB
Einfach einstecken - fertig!! **DM 995,--**

**Achtung Preisänderung!**

**Power CAD Plus**

Das **Grafiktableau** der Superlative!
Mit diesem Eingabegerät können Sie nach der Installation auf Tastatur, Joystick, Maus oder Trackball verzichten, denn dies alles und noch vieles weiteres ist bereits in dem Tableau integriert (siehe AMSTRAD International Nr. 8/88). **DM 998,--**

**Neu • Neu • Neu**

**TurboCAD**

**TurboCAD** von PINK SOFTWARE ist ein allein-stehendes Werkzeug zum Erstellen von Zeichnungen, Grafiken und technischen Spezifikationen. Diese benutzerfreundliche Software wurde unter Mithilfe von Fachleuten der Konstruktion und Technik auf den neuesten Stand der Zeichentechnik gebracht.
Im Lieferumfang enthalten ist ein ausführliches deutsches Handbuch.
**Info anfordern** (bitte schriftlich)
**nur DM 348,--**

**Public Domain Software**
**MS-DOS**

Mehr als 700 Disk. sofort lieferbar!
**Liste anfordern !**
pro Diskette nur **8,-- DM**

**PD-10er Blöcke**

Block 1
- Textverarbeitung
- Flightsimulator
- PC File III
- Printer Utilities
- Basic Progr. Utilit.
- Wertpepier/Finanzen
- Mini Assembler
- Programm Control
- "Crossref"
- " Startrek"

Block 2
- Datenbankorganisation
- Brasef + Make
- Bowling Secretary
- Finance
- Math und Stat
- Print Grafik I+II
- Basic Translator
- ESIE künstl. Intelligenz
- Wizard´s Castle Game
- Packman and Newtrek

Block 3
- PC-Write
- Poster
- Tabellenkakulation
- Spiele deutsch
- Prolog
- Datenbank
- Finance
- DOS Tutor, deutsch
- Monopoly
- Horoskop

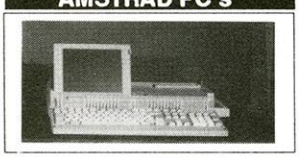
Block 4
- Drucker-Utilities
- Musik
- Touchtype
- Applikation
- PC-Calc
- Infobase Datenbank
- PC-ZAP
- Hilfsprogramme
- Drucker Utilities
- Basic Utilities

je Block nur **68,-- DM**

**Nützliches Zubehör**

| | |
|---|---|
| **Handy Scanner** | 798,-- |
| ... mit Texterkennung | 848,-- |
| Laufwerk 51/4", 3,5" | je 398,-- |
| Aufrüstsatz 256 auf 512K | 248,-- |
| Aufrüstsatz 512 auf 640K | 178,-- |
| Hercules Kit für 1512 | 348,-- |
| PC-Maus mit Software | 98,-- |
| Gleitmatte für PC-Maus | 19,80 |
| Videodigitizer PC | 398,-- |
| Lightpen PC | 98,-- |

**AMSTRAD PC's**



| PC 1640: | | | |
|---|---|---|---|
| MD/SD | 1.699,-- | CD/DD | 2.399,-- |
| MD/DD | 1.999,-- | CD/HD20 | 2.999,-- |
| MD/HD20 | 2.599,-- | ECD/SD | 2.699,-- |
| CD/SD | 2.099,-- | ECD/DD | 2.999,-- |
| | | ECD/HD20 | 3.599,-- |

| PC 1512: | |
|---|---|
| MD/SD | 1.299,-- |
| MD/DD | 1.599,-- |
| CD/SD | 1.699,-- |
| CD/DD | 1.999,-- |

**Portable PC 512:**

| | |
|---|---|
| 3,5" Laufw. | 1.699,-- |
| 2 Laufw. | 1.999,-- |

**Drucker:**

| | |
|---|---|
| LQ 3500 | 899,-- |
| LQ 5000 | 1.399,-- |

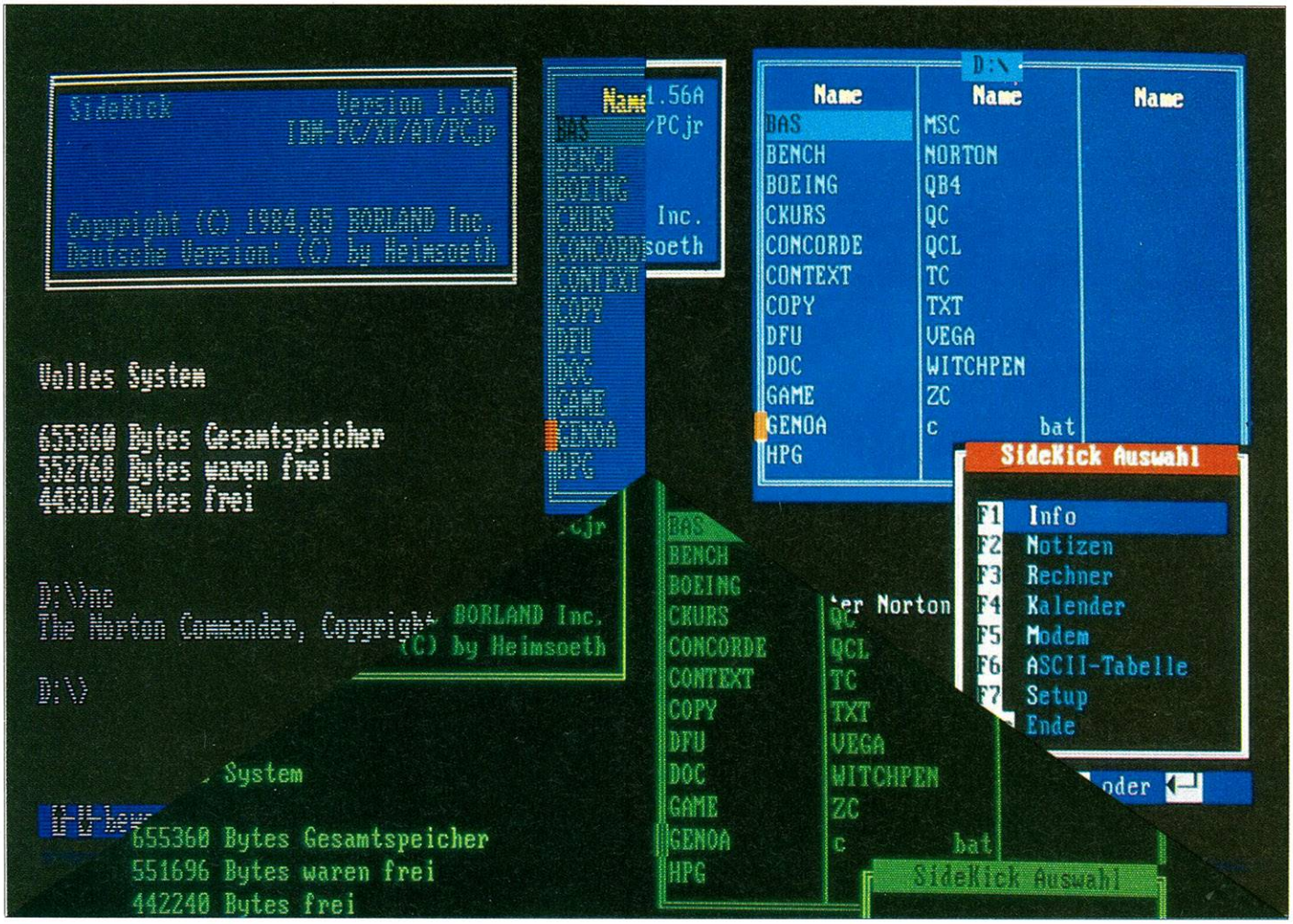
**Info anfordern!**

Karl-Heinz Weeske • Potsdamer Ring 10 • 7150 Backnang • Telex 724410 weeba d • Kreisspark. Backnang (BLZ 60250020) 74397 • FAX 60077

**weeske**
COMPUTER-ELEKTRONIK

Zahlung im Inland per Nachnahme, im Ausland per Scheck erwünscht. Versandkostenpauschale (Inland 6,80 DM / Ausland 16,80 DM). Bei Infoanforderung bitte DM 3,– in Briefmarken beilegen.

**Tel.: 07191/1528-29 od. 60076**




# Alles über Grafikkarten

**Für den PC gibt es, wenn man von dem noch recht jungen VGA-Standard einmal absieht, drei wichtige Grafikkartentypen, die sich in der Ansteuerung erheblich voneinander unterscheiden. Wer die Unterschiede kennenlernen oder Software schreiben möchte, die mit allen drei Grafikstandards zusammenarbeitet, findet in diesem Beitrag die entsprechenden Informationen.**

Die erste Bildschirmkarte, die für den IBM-PC erhältlich war, ist der sogenannte Monochromadapter, der oft auch mit MDA abgekürzt wird. Der Monochromadapter wurde 1981 zusammen mit dem ersten IBM-PC vorgestellt. Die Leistungsfähigkeit des MDA ist allerdings sehr beschränkt, denn er ist nicht grafikfähig, das heißt, der Monochromadapter kann nur die Buchstaben, Sonderzeichen und Blockgrafikzeichen aus dem IBM-Zeichensatz darstellen. Dafür ist die Textqualität des MDA ausgezeichnet, denn ein Zeichen besteht aus einer Matrix von 9x14 Bildpunkten. Insgesamt passen 80x25 Zeichen auf eine Bildschirmseite. Dies ist allerdings auch das einzige technische Datum des Monochromadapters, das bis heute für Grafikkarten Standard geblieben ist.

Der offizielle Bildschirmspeicher des Monochromadapters ist 4000 Byte groß und reicht von B0000hex bis B0F9Fhex. Aus technischen Gründen ist der Bildschirmspeicher in der Realität aber nicht 4000 Byte, sondern 4 KByte, also 4096 Byte groß. Die überflüssigen Bytes des Bildschirmspeichers lassen sich hervorragend dazu benutzen, um festzustellen, ob im PC ein Monochromadapter eingebaut ist.

Der Programmierer muß nur durch wiederholte Schreib- und Lesezugriffe auf diesen Speicherbereich prüfen, ob Speicherbausteine vorhanden sind oder nicht.

Der Anwender bemerkt eine solche Testroutine nicht, denn weil der Monochromadapter den an sich überflüssigen Bereich des Speichers nicht benutzt, ändert sich auch der Bildschirminhalt nicht. In GW-Basic läßt sich zum Beispiel folgende Routine verwenden, um zu testen, ob ein Monochromadapter vorhanden ist:

```
1000 def seg=&Hb000
1010 mda=1
1020 for a=0 to 255
1030 poke &Hfa1,a
1040 if peek(&Hfa1)<>a then mda=0
1050 next a
```

Dieses kurze Unterprogramm liefert in der Variable »mda« den Wert 1 zurück, wenn der Speichertest keinen Fehler ergeben hat, andernfalls den Wert 0.

Das aufgeführte Programm eignet sich allerdings nur für die Unterscheidung zwischen CGA- und MDA-Karte. Sobald eine EGA-Karte im PC installiert ist, liefert die Routine den Wert 1 zurück, weil der 256 KByte große Bildschirmspeicher einer EGA-Karte auch diesen Bereich überdeckt.

Im Bildschirmspeicher des Monochromadapters sind für jede Zeichenposition auf dem Bildschirm zwei Byte zuständig. Das erste Byte enthält den ASCII-Wert des jeweiligen Zeichens und das zweite Byte das Attribut für dieses Zeichen, das heißt die Angabe, ob das Zeichen normal, leuchtend, invers, blinkend oder unterstrichen dargestellt wird.

## Ein Zeichen auf dem Bildschirm ist beim Monochromadapter durch zwei Byte im Bildschirmspeicher repräsentiert

Tabelle 1 zeigt den Aufbau des Attributbytes. Die Funktion von Bit 7 des Attributbytes ist vom Zustand des Bit 5 im Steuerregister des Monochromadapters (Portadresse 3B8hex) abhängig. Ist das Bit 5 im Steuerregister gesetzt, so gibt Bit 7 des Attributbytes an, ob das Zeichen normal oder blinkend dargestellt wird. Ist Bit 5 des Steuerregisters dagegen gelöscht, so bestimmt das Bit 7 des Attributbytes, ob der Hintergrund normal oder leuchtend dargestellt wird.
Eine Textzeile von 80 Zeichen belegt aufgrund des zusätzlichen Attributbytes für jedes Zeichen nicht 80, sondern 160 Byte. Die relative Adresse einer beliebigen Bildschirmposition (relativ zum Anfang des Bildschirmspeichers) errechnet sich nach folgender Formel:

```
ADR=160 x (zeile - 1) + 2 x (spalte - 1)
```

Die relative Adresse des Attributs läßt sich dementsprechend mit

```
ADR=160 x (zeile - 1) + 2 x (spalte - 1) + 1
```

berechnen, wobei die zulässigen Werte für »spalte« zwischen 1 und 80 und die Werte für »zeile« zwischen 1 und 25 liegen müssen. Die relativen Adressen lassen sich in absolute Adressen umrechnen, indem Sie jeweils die Startadresse des Bildschirmspeichers (B0000hex) addieren.
Als Weiterentwicklung des Monochromadapters konstruierte die Firma Hercules die heute weit verbreitete Hercules-Grafikkarte, die im Gegensatz zum Monochromadapter grafikfähig ist. Diese Karte enthält einen Schaltungsteil, der mit dem Monochromadapter nahezu identisch ist und ersetzt den Monochromadapter daher vollständig. Zusätzlich ist die Hercules-Grafikkarte jedoch auch mit 64 KByte Bildschirmspeicher und weiterer Elektronik ausgestattet.
So wurde erreicht, daß die Herculeskarte einerseits den Textmodus des Monochromadapters beherrscht und dazu 100 Prozent kompatibel ist, und andererseits auch in einen echten Grafikmodus umgeschaltet werden kann, der eine Auflösung von 720x348 Bildpunkten – allerdings nur einfarbig – erlaubt.

Außerdem besitzt die Hercules-Grafikkarte gegenüber dem Monochromadapter zwei zusätzliche Register, die die Konfiguration und den Betriebsmodus der Hercules-Grafikkarte bestimmen. Über das Konfigurationsregister läßt sich der insgesamt 64 KByte große Bildschirmspeicher der Hercules-Grafikkarte in drei Schritten ein- und ausblenden.

Die drei zulässigen Größen für den Bildschirmspeicher werden mit »Diag«, »Half« und »Full« bezeichnet.

In der Diag-Konfiguration sind nur 4 KByte des Bildschirmspeichers aktiv. In diesem Modus verhält sich die Hercules-Grafikkarte wie ein echter Monochromadapter. Daher ist bei dieser Konfiguration auch nur die Textdarstellung möglich.

## Bei der Hercules-Grafikkarte läßt sich der Bildschirmspeicher in drei verschiedenen Größen konfigurieren

Die Half-Konfiguration aktiviert, wie der Name schon sagt, die Hälfte des Bildschirmspeichers, also 32 KByte (Adresse B0000 bis B7FFFhex). Um die Herculeskarte im Grafikmodus zu betreiben, ist zumindest die Half-Konfiguration erforderlich.

Die Half-Konfiguration ist dann sinnvoll, wenn die Herculeskarte zusammen mit einer CGA-Farbgrafikkarte in einem PC installiert ist. Andernfalls würden sich die oberen 32 KByte der Herculeskarte mit dem Bildschirmspeicher der CGA-Farbgrafikkarte überschneiden.

Die Full-Konfiguration blendet schließlich den gesamten Bildschirmspeicher der Herculeskarte ein (Adresse B0000 bis BFFFFhex).

Weil eine komplette Grafikseite der Herculeskarte knapp 32 KByte des Bildschirmspeichers benötigt, erhalten Sie mit der Full-Konfiguration zwei Grafikseiten, zwischen denen Sie über Bit 7 des Modusregisters umschalten.

Tabelle 2 zeigt die Portadressen des Monochromadapters und der Hercules-Grafikkarte. Das Konfigurationsregister läßt sich über die Portadresse 3BFhex ansprechen. (Nur Schreibzugriffe haben eine Wirkung.) Bit 0 und Bit 1 dieses Registers bestimmen die Konfiguration des Bildschirmspeichers:

**Bit 0 = 0**

Wenn dieses Bit nicht gesetzt ist, ist es auch nicht möglich, das Grafikmodus-Bit 1 im Steuerregister der Herculeskarte zu setzen. Nach dem Einschalten des PC ist das Bit 0 des Konfigurationsregisters immer gelöscht.

**Bit 0 = 1**

Ist Bit 0 dagegen gesetzt, so läßt sich das Grafikmodus-Bit setzen.

**Bit 1 = 0**

Wenn dieses Bit gelöscht ist, ist die zweite Bildschirmseite der Hercules-Grafikkarte, also der Bildschirmspeicher von Adresse B8000 bis BFFFFhex ausgeblendet. Außerdem bleibt das Setzen des Seitenauswahl-Bit 7 im Steuerregister der Herculeskarte ohne Wirkung. Nach dem Einschalten des PC ist das Bit 1 des Konfigurationsregisters immer gelöscht.

**Bit 1 = 1**

Ist Bit 1 dagegen gesetzt, so ist auf der Herculeskarte der Speicherbereich von B8000 bis BFFFFhex aktiviert, das heißt, die zweite Grafikseite läßt sich nun durch Setzen des Seitenauswahl-Bit 7 einschalten.
Die drei zuvor angesprochenen Bildschirmspeicher-Konfigurationen »Diag«, »Half« und »Full« lassen sich damit durch einen einfachen Out-Befehl unter GW-Basic auswählen. Der Befehl

```
out &H3BF,0
```

aktiviert die Diag-Konfiguration, mit

```
out &H3BF,1
```

wird die Half-Konfiguration eingeschaltet und

```
out &H3BF,3
```

bewirkt die Wahl der Full-Konfiguration.
Über das Konfigurationsregister wird allerdings lediglich die Größe des Bildschirmspeichers bestimmt, so daß die Voraussetzung zum Umschalten in den Grafikmodus gegeben sind. Um den Grafikmodus der Herculeskarte dann endgültig zu aktivieren, müssen Sie folgende Aktionen durchführen:

## Zum Auswählen der Speicherkonfiguration für die Hercules-Grafikkarte genügt ein einziger Basic-Befehl

1. Mindestens 32 KByte Bildschirmspeicher einblenden, das heißt die Half- oder Full-Konfiguration wählen.
2. Das Grafikmodus-Bit 1 im Steuerregister der Hercules-Grafikkarte setzen.
3. Die internen Register des Video-Controllers 6845 mit neuen Werten laden.
4. Den Bildschirmspeicher löschen.

An dieser Stelle ist es nun erforderlich, die Funktion der einzelnen Bits im Steuerregister der Hercules-Grafikkarte (Portadresse 3B8hex), die

teilweise schon mehrmals erwähnt wurden, genauer zu erläutern.

**Bit 0**
Nicht verwendet.

**Bit 1 = 0**
Wenn dieses Bit gelöscht ist, interpretiert die Hercules-Grafikkarte die Daten im Bildschirmspeicher als Textzeichen und führt das erste Byte dem Zeichengenerator zu, das zweite dem Attributcontroller, das dritte wieder dem Zeichengenerator und so weiter. Nach dem Einschalten des PC ist das Bit 1 des Steuerregisters immer gelöscht.

**Bit 1 = 1**
Ist Bit 1 dagegen gesetzt, so interpretiert die Hercules-Grafikkarte die Daten des Bildschirmspeichers als Grafikdaten und setzt Zeichengenerator sowie Attributcontroller außer Betrieb. Für den Grafikmodus der Herculeskarte muß das Bit 1 gesetzt sein. Dies bedeutet aber nicht, daß die Herculeskarte durch einfaches Setzen des Bit 1 in den Grafikmodus umgeschaltet wird. Vielmehr muß auch der Video-Controller 6845 jedesmal mit neuen Registerwerten geladen werden, wenn das Bit 1 seinen Wert ändert. Bit 1 läßt sich nur dann setzen, wenn auch das Bit 0 im Konfigurationsregister gesetzt ist.

**Bit 2**
Nicht verwendet.

## Über das Steuerregister der Hercules-Grafikkarte lassen sich die wichtigsten Parameter einstellen

**Bit 3 = 0**
Wenn dieses Bit gelöscht ist, wird die Bildschirmanzeige unterdrückt und der Monitor bleibt dunkel. Nach dem Einschalten des PC ist das Bit 3 immer gelöscht. Erst nachdem das BIOS die Register des Video-Controllers 6845 mit den erforderlichen Parametern initialisiert hat, setzt es das Bit 3 und schaltet so die Bildschirmanzeige ein. Das Bit 3 wird auch beim Umschalten zwischen Text- und Grafikmodus gelöscht, weil während der Neuprogrammierung des Video-Controllers undefinierte Signale am Ausgang der Hercules-Grafikkarte entstehen können.

**Bit 3 = 1**
Sobald das Bit 3 gesetzt ist, wird die Bildschirmanzeige aktiviert.

**Bit 4**
Nicht verwendet.

**Bit 5 = 0**
Wenn dieses Bit gelöscht ist, lassen sich auch Zeichen mit Blink-Attribut nicht blinkend darstellen.

**Bit 5 = 1**
Ist Bit 5 dagegen gesetzt, so werden alle mit einem Blink-Attribut verse-

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Funktion |
|---|---|---|---|---|---|---|---|---|---|
| | B | 0 | 0 | 0 | I | 0 | 0 | 0 | keine Anzeige |
| | B | 0 | 0 | 0 | I | 0 | 0 | 1 | unterstrichen |
| | B | 0 | 0 | 0 | I | 1 | 1 | 1 | normale Anzeige |
| | B | 1 | 1 | 1 | I | 0 | 0 | 0 | inverse Anzeige |

I = 0: normale Darstellung
I = 1: leuchtende Darstellung
B = 0: normale Darstellung/normaler Hintergrund
B = 1: blinkende Darstellung/leuchtender Hintergrund

**Tabelle 1. Der Aufbau des Attributbytes im Textmodus des Monochromadapters und der Hercules-Grafikkarte**

| Adresse (hex) | Funktion |
|---|---|
| 3B0 | nicht verwendet |
| 3B1 | nicht verwendet |
| 3B2 | nicht verwendet |
| 3B3 | nicht verwendet |
| 3B4 | Indexregister des Video-Controllers |
| 3B5 | Datenregister des Video-Controllers |
| 3B6 | nicht verwendet |
| 3B7 | nicht verwendet |
| 3B8 | Steuerregister des Bildschirmadapters |
| 3B9 | Lightpen-Flip-Flop setzen (*) |
| 3BA | Statusregister des Bildschirmadapters |
| 3BB | Lightpen-Flip-Flop löschen (*) |
| 3BC | Datenregister der Druckerschnittstelle |
| 3BD | Statusregister der Druckerschnittstelle |
| 3BE | Steuerregister der Druckerschnittstelle |
| 3BF | Konfigurationsregister (**) |

(*) IBM bezeichnet diese Adressen als reserviert.
(**) Das Konfigurationsport ist nur auf einer Hercules-Grafikkarte vorhanden. Durch einen Schreibbefehl auf diese Adresse kann der Programmierer den Bildschirmspeicher teilweise ein- oder ausblenden.

**Tabelle 2. Der Adreßbereich eines Monochromadapters und einer Hercules-Grafikkarte umfaßt 16 Portadressen**

| Intensität | Rot | Grün | Blau | Farbe |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | Schwarz |
| 0 | 0 | 0 | 1 | Blau |
| 0 | 0 | 1 | 0 | Grün |
| 0 | 0 | 1 | 1 | Kobaltblau |
| 0 | 1 | 0 | 0 | Rot |
| 0 | 1 | 0 | 1 | Violett |
| 0 | 1 | 1 | 0 | Braun |
| 0 | 1 | 1 | 1 | Weiß |
| 1 | 0 | 0 | 0 | Grau |
| 1 | 0 | 0 | 1 | Hellblau |
| 1 | 0 | 1 | 0 | Hellgrün |
| 1 | 0 | 1 | 1 | Hellkobaltblau |
| 1 | 1 | 0 | 0 | Hellrot |
| 1 | 1 | 0 | 1 | Hellviolett |
| 1 | 1 | 1 | 0 | Gelb |
| 1 | 1 | 1 | 1 | Weiß (intensiv) |

**Tabelle 3. Die CGA-Karte kann durch Kombination der drei Grundfarben und einem Intensitätssignal 16 verschiedene Farben darstellen**

henen Zeichen auch tatsächlich blinkend dargestellt.

**Bit 6**

Nicht verwendet

**Bit 7 = 0**

In der Diag- oder Half-Konfiguration ist dieses Bit immer gelöscht.

**Bit 7 = 1**

Befindet sich die Hercules-Grafikkarte dagegen in der Full-Konfiguration, so legt das Bit 7 fest, ob die erste Bildschirmseite (Startadresse B0000hex) oder die zweite Bildschirmseite (Startadresse B8000hex) aktiv ist. Nur in der Full-Konfiguration der Hercules-Grafikkarte läßt sich der Wert des Bit 7 verändern.

Wie Sie bereits gelesen haben, reicht es nicht aus, das Grafikmodus-Bit im Steuerregister der Herculeskarte zu setzen, um in den Grafikmodus umzuschalten. Vielmehr muß auch der Video-Controller 6845 auf der Hercules-Grafikkarte neu programmiert werden.

Der Video-Controller ist das Herz jedes Bildschirmadapters. Er liest die im Bildschirmspeicher abgelegten Daten ein und erzeugt daraus ein Videosignal.

Im Textmodus hilft dem Video-Controller zusätzlich ein Zeichengenerator, oft auch als Charaktergenerator bezeichnet, bei der Erzeugung der Bildpunktmatrizen für die einzelnen Zeichen. Ein Zeichengenerator besteht im wesentlichen aus einem Festwertspeicher (ROM oder EPROM), der die im Bildschirmspeicher enthaltenen Zeichencodes in Bildpunkte umsetzt.

## Die Arbeitsweise des Video-Controllers ist im Text- und Grafikmodus sehr unterschiedlich

Im Grafikmodus liest der Video-Controller die Daten des Bildschirmspeichers und gibt diese direkt als Videosignal aus, weil jedes Bit im Bildschirmspeicher einen Bildpunkt oder die Farbe des Bildpunktes auf dem Monitor repräsentiert. Eine Umsetzung der Daten aus dem Bildschirmspeicher in Bildpunkte durch den Zeichengenerator ist deshalb nicht erforderlich.

Sowohl der Monochromadapter, als auch die Hercules-Grafikkarte und die CGA-Farbgrafikkarte verwenden als Video-Controller den Motorola-Baustein 6845. Dieser Baustein läßt sich sehr vielseitig einsetzen, denn er enthält 18 interne Register, über die sich die Bilderzeugung in weiten Grenzen beeinflussen läßt.

Um auf eines der internen Register des Video-Controllers zuzugreifen, müssen Sie zuerst in das Indexregister des 6845 (Portadresse 03B4hex)

| Register | Standardwerte DA/HGA (Text) | HGA (Grafik) | CGA (80x25) | CGA (40x25) | CGA (Grafik) | Funktion |
|---|---|---|---|---|---|---|
| 0 | 97 | 53 | 113 | 56 | 56 | Anzahl Zeichen horizontal insgesamt |
| 1 | 80 | 45 | 80 | 40 | 40 | Anzahl der horizontalen Zeichen, die angezeigt werden |
| 2 | 82 | 46 | 90 | 45 | 45 | (zeitliche) Position des HSYNC-Signals |
| 3 | 15 | 7 | 10 | 10 | 10 | Synchronisationsbreite |
| 4 | 25 | 91 | 31 | 31 | 127 | Anzahl Zeilen insgesamt |
| 5 | 6 | 2 | 6 | 6 | 6 | Feineinstellung für Register 4 |
| 6 | 25 | 87 | 25 | 25 | 100 | Anzahl der Zeilen, die angezeigt werden |
| 7 | 25 | 87 | 28 | 28 | 112 | (zeitliche) Position des VSYNC-Signals |
| 8 | 2 | 2 | 2 | 2 | 2 | Interlace (Zeilensprungmodus) ein/aus |
| 9 | 13 | 3 | 7 | 7 | 1 | Höchste Rasterzeilen-Adresse |
| 10 | 11 | 0 | 6 | 6 | 6 | Start-Rasterzeile des Cursors |
| 11 | 12 | 0 | 7 | 7 | 7 | End-Rasterzeile des Cursors |
| 12 | xx | xx | xx | xx | xx | Höherwertiges Byte der Startadresse des Bildschirmspeichers |
| 13 | xx | xx | xx | xx | xx | Niederwertiges Byte der Startadresse des Bildschirmspeichers |
| 14 | xx | xx | xx | xx | xx | Höherwertiges Byte der Adresse des Cursors relativ zum Anfang des Bildschirmspeichers |
| 15 | xx | xx | xx | xx | xx | Niederwertiges Byte der Adresse des Cursors relativ zum Anfang des Bildschirmspeichers |
| 16 | xx | xx | xx | xx | xx | Höherwertiges Byte der Adresse der Bildschirmposition der letzten Lightpen-Position |
| 15 | xx | xx | xx | xx | xx | Niederwertiges Byte der Adresse der Bildschirmposition der letzten Lightpen-Position |

(Alle Werte dezimal)

**Tabelle 4. Die 18 internen Register des Video-Controllers 6845 erlauben eine flexible Programmierung der Bildschirmausgabe**

| Farben | Modus | Monitor | Grafikauflösung | Textauflösung | Karte |
|---|---|---|---|---|---|
| – | Text | CD,ECD | 320x200 | 40x25 | CGA,EGA |
| 16 | Text | CD,ECD | 320x200 | 40x25 | CGA,EGA |
| – | Grafik | CD,ECD | 320x200 | 40x25 | CGA,EGA |
| 4 | Grafik | CD,ECD | 320x200 | 40x25 | CGA,EGA |
| 16 | Grafik | CD,ECD | 320x200 | 40x25 | EGA |
| – | Text | CD,ECD | 640x200 | 80x25 | CGA,EGA |
| 16 | Text | CD,ECD | 640x200 | 80x25 | CGA,EGA |
| – | Grafik | CD,ECD | 640x200 | 80x25 | CGA,EGA |
| 16 | Grafik | CD,ECD | 640x200 | 80x25 | EGA |
| 16/64 | Text | ECD | 640x350 | 80x25 | EGA |
| 16/64 | Text | ECD | 640x350 | 80x43 | EGA |
| – | Grafik | MD | 640x350 | 80x25 | EGA |
| 16/64 | Grafik | ECD | 640x350 | 80x25 | EGA |
| – | Text | MD | 720x350 | 80x25 | Monochromadapter,EGA |
| – | Grafik | MD | 720x350 | 80x25 | Hercules |

CD = Standard-Farbmonitor, der wegen seiner Horizontalfrequenz von 15,75 kHz nur die CGA-Modi und die erweiterten CGA-Modi der EGA-Karte (also alle EGA-Modi mit einer Auflösung von 640x200 oder 320x200) beherrscht
ECD = EGA-Monitor, der selbsttätig die richtige Horizontalfrequenz von 15,75 kHz (für CGA-Modi) oder 21,85 kHz (für EGA-Modi) einstellt und daher für alle CGA- und EGA-Modi (abgesehen von dem monochromen EGA-Modus mit 640x350 Bildpunkten) geeignet ist
MD = Monochrom-Monitor, der mit einer Horizontalfrequenz von 18,43 kHz für den Anschluß eines Monochromadapters oder einer Hercules-Grafikkarte und für die monochromen EGA-Modi geeignet ist

**Tabelle 5. Die verschiedenen Videomodi der PC-Grafikkarten unterscheiden sich in Auflösung und Farbenvielfalt**

| Adresse (hex) | Funktion |
|---|---|
| 3D0 | nicht verwendet |
| 3D1 | nicht verwendet |
| 3D2 | nicht verwendet |
| 3D3 | nicht verwendet |
| 3D4 | Indexregister des Video-Controllers |
| 3D5 | Datenregister des Video-Controllers |
| 3D6 | nicht verwendet |
| 3D7 | nicht verwendet |
| 3D8 | Modusauswahlregister |
| 3D9 | Farbauswahlregister |
| 3DA | Grafikstatusregister |
| 3DB | Lightpen-Flip-Flop löschen |
| 3DC | Lightpen-Flip-Flop setzen |
| 3DD | nicht verwendet (*) |
| 3DE | nicht verwendet (*) |
| 3DF | nicht verwendet (*) |

* Diese drei Portadressen werden beim Schneider PC 1512 für den speziellen Grafikmodus 2 mit 640x200 Bildpunkten in 16 Farben verwendet.

**Tabelle 6. Die CGA-Farbgrafikkarte verfügt wie der Monochromadapter und die Herculeskarte über 16 Portadressen**

die Nummer des gewünschten Registers, also eine Zahl zwischen 0 und 17, schreiben. Danach können Sie über das Datenregister (Portadresse 03B5hex) den Inhalt des ausgewählten Registers einlesen oder einen neuen Wert in das Register schreiben.
Vor dem wahllosem Ausprobieren von neuen Werten für die verschiedenen Register ist jedoch dringend abzuraten, weil bei Extremwerten Beschädigungen am Monitor nicht auszuschließen sind.
Die Werte in den internen Registern des Video-Controllers sind für Text- und Grafikmodus zum Teil sehr unterschiedlich und variieren auch von Grafikkartentyp zu Grafikkartentyp. Tabelle 3 zeigt die Standardwerte der Register für die verschiedenen Modi von Monochromadapter (MDA), Hercules-Grafikkarte (HGA) und CGA-Farbgrafikkarte. Nach dem Einschalten des PC schreibt das BIOS diese Werte für den per Dipschalter gewählten Videomodus in die Register des Video-Controllers.
Doch zurück zur Programmierung des Grafikmodus für die Herculeskarte. In der Spalte »HGA (Grafik)« sind die Werte aufgeführt, die Sie in die ersten zwölf Register des Video-Controllers schreiben müssen, wenn Sie den Grafikmodus der Herculeskarte auswählen möchten.
Nachdem Sie diese Werte in die Register des Video-Controllers geschrieben haben, müssen Sie nur noch eine Routine zum Setzen und Prüfen von Bildpunkten programmieren, weil die Hercules-Grafikkarte leider keine BIOS-Erweiterung besitzt, die für den Video-Interrupt 10hex entsprechende Funktionen zur Verfügung stellt. Auch können Sie die bereits bestehenden Funktionen 12 und 13 dieses Interrupts zum Setzen und Prüfen von Bildpunkten nicht verwenden, weil diese speziell für die CGA-Farbgrafikkarte geschrieben wurden.

## Mit einer Formel läßt sich berechnen, welches Bit im Bildschirmspeicher für welchen Bildpunkt zuständig ist

Die Routine zum Setzen eines Bildpunktes muß aus dessen x/y-Koordinate die Adresse der zugehörigen Speicherzelle im Bildschirmspeicher und die zuständige Bitposition innerhalb dieser Speicherzelle berechnen. Wenn nun das berechnete Bit gesetzt wird, schaltet die Hercules-Grafikkarte den entsprechenden Bildpunkt hell, und wird das Bit zurückgesetzt, so wird auch der Bildpunkt wieder dunkel.
Das im Bildschirmspeicher zuständige Bit für eine bestimmte Bildschirmkoordinate läßt sich berechnen, weil der Bildschirmspeicher im Grafikmodus regelmäßig, allerdings etwas ungewöhnlich, organisiert ist. Der 32 KByte große Bildschirmspeicher ist nämlich in vier Teile zu je 8 KByte aufgeteilt, wobei der erste Teil die Informationen für die ohne Rest durch vier teilbaren Bildschirmzeilen enthält, der zweite Teil die Informationen für die Bildschirmzeilen, die beim Teilen durch vier einen Rest von Eins ergeben, der dritte Teil die Informationen für die Bildschirmzeilen mit einem Rest von zwei und der vierte Teil die Informationen für die Bildschirmzeilen mit einem Rest von drei.
Die relative Bildschirmspeicheradresse für einen bestimmten Bildpunkt berechnet sich nach der Formel

```
ADR=2000hex x (Y mod 4) + 90 x int(Y/4) +
int(X/8)
```

wobei die Variablen X (Wertebereich 0 bis 719) und Y (Wertebereich 0 bis 347) die Koordinate des gesuchten Bildpunktes angeben. Die absolute Bildschirmspeicheradresse erhalten Sie, indem Sie zu der relativen Adresse die Startadresse des Bildschirmspeichers, also den Wert B0000hex oder B8000hex addieren. Das für den Bildpunkt zuständige Bit innerhalb der berechneten Speicherstelle errechnen Sie aus der Formel

```
BIT=7 - (X mod 8)
```

Der Hercules-Standard für monochrome Grafikkarten ist übrigens neben dem EMS-Standard, der von Lotus, Intel und Microsoft entwickelt wurde, die einzige Norm im Bereich der IBM-kompatiblen Computer, die nicht von IBM stammt und dennoch weite Verbreitung erlangt hat.
Die Tatsache, daß die Herculeskarte das Produkt eines Fremdanbieters ist, hat allerdings auch dazu geführt, daß weder das BIOS, noch MS-DOS oder GW-Basic den Grafikmodus der Herculeskarte unterstützen. Besonders die Besitzer von Hercules-kompatiblen Karten sind benachteiligt, denn Hercules liefert zu jeder Karte eine Sammlung von nützlichen Routinen zum Ansteuern des Grafikmodus mit. Der Großteil der Nachbauten wird dagegen ohne Software ausgeliefert. Daher müssen Sie sich in 90 Prozent der Fälle eigene Routinen zum Ansteuern des Hercules-Grafikmodus schreiben. In der DOS International, Ausgabe 2/'88 wurde bereits beschrieben, wie Sie Ihre Hercules-Grafikkarte von GW-Basic aus ansprechen.

| Befehl | Videomodus |
|---|---|
| SCREEN 0 | Textmodus (Anzahl der Spalten bleibt unverändert) |
| SCREEN 1 | Grafikmodus mit 320x200 Bildpunkten in vier Farben (nur CGA und EGA) |
| SCREEN 2 | Grafikmodus mit 640x200 Bildpunkten einfarbig (nur CGA und EGA) |
| SCREEN 7 | Erweiterte CGA-Grafik mit 320x200 Bildpunkten in 16 Farben (nur EGA) |
| SCREEN 8 | Erweiterte CGA-Grafik mit 640x200 Bildpunkten in 16 Farben (nur EGA) |
| SCREEN 9 | EGA-Grafikmodus mit 640x350 Bildpunkten in 4 beziehungsweise 16 aus 64 Farben (*) |
| SCREEN 10 | Monochrome EGA-Grafik mit 640x200 Bildpunkten (**) |

(*) Dieser Modus ist nur zulässig, wenn eine EGA-Karte mit EGA-Monitor installiert wurde. Die Anzahl der maximal darstellbaren Farben ist abhängig von der Größe des Bildschirmspeichers der EGA-Karte. Heute sind fast alle EGA-Karten mit 256 KByte Bildschirmspeicher ausgerüstet, und können daher 16 Farben von 64 möglichen Farben gleichzeitig anzeigen.

(**) Dieser Modus ist für eine EGA-Karte gedacht, an die ein Monochrom-Monitor mit 18,43 kHz Horizontalfrquenz angeschlossen ist.

**Tabelle 7. Mit GW-Basic lassen sich über den Screen-Befehl sieben verschiedene Videomodi setzen**

## Mit Hilfe eines CGA-Emulators können Sie auch auf der Herculeskarte CGA-Grafiken darstellen

Eine weitere Möglichkeit, Ihre Herculeskarte im Grafikmodus zu benutzen, besteht in der Verwendung von sogenannte CGA-Emulatoren, wie zum Beispiel das Programm »Multigraph II« von Kirschbaum Software. Derartige Programme emulieren die Grafikmodi einer CGA-Farbgrafikkarte auf der Herculeskarte. So lassen sich Spiele oder die Grafikbefehle von GW-Basic und anderen Programmiersprachen, die für die CGA-Karte gedacht sind, auch auf der Herculeskarte verwenden.
1983, zwei Jahre nach Einführung des IBM-PC, stellte IBM eine Farbgrafikkarte für den PC vor, die heute als CGA-Karte weite Verbreitung erlangt hat. Die CGA-Farbgrafikkarte hat den Vorteil, daß sie relativ preiswert ist und sowohl Text- als auch Grafikdarstellung in Farbe bietet.
Zwar wird die CGA-Farbgrafikkarte wegen der gesunkenen Hardwarepreise zunehmend von der EGA-Karte abgelöst, dennoch spielt sie immer noch eine wichtige Rolle, insbesondere im Heimbereich.
Die CGA-Karte kennt prinzipiell vier Videomodi, zwei für Text und zwei für Grafik:

- 40x25 Zeichen in 16 Farben
- 80x25 Zeichen in 16 Farben
- 320x200 Bildpunkte in 4 Farben
- 640x200 Bildpunkte in 2 Farben

Tabelle 4 zeigt, wie die CGA-Karte durch Kombination der drei Grundfarben und ein zusätzliches Intensitätssignal 16 verschiedene Farben erzeugt.

Zusätzlich läßt sich die Farbdarstellung in den drei ersten Modi auch abstellen, wodurch sich drei weitere Videomodi ergeben. Tabelle 5, die auch den Monochromadapter, die Hercules-Grafikkarte und die EGA-Karte berücksichtigt, zeigt eine Aufstellung aller Videomodi.
Eine CGA-Farbgrafikkarte ist mit 16 KByte Bildschirmspeicher ausgerüstet, der im Adreßbereich von B8000 bis BBFFFhex liegt. Weil die CGA-Karte von IBM entwickelt wurde, wird sie auch vom BIOS direkt unterstützt. Der Assembler-Programmierer kann über den Video-Interrupt 10hex eine Vielzahl von Funktionen auswählen.
In den Textmodi ist der Bildschirmspeicher der CGA-Farbgrafikkarte analog zum Speicher des Monochromadapters organisiert. Für jede Zeichenposition auf dem Bildschirm sind zwei Byte reserviert. Das erste Byte enthält den Zeichencode für die Bildschirmposition und das zweite Byte das Attribut für das Zeichen.

## Auch die CGA-Farbgrafikkarte arbeitet in den Textmodi mit je einem Byte für das Zeichen und das Attribut

Das Attributbyte setzt sich bei der CGA-Farbgrafikkarte wie folgt zusammen:

| Bit | Funktion |
|---|---|
| 0 | blaues Zeichen |
| 1 | grünes Zeichen |
| 2 | rotes Zeichen |
| 3 | leuchtendes Zeichen |
| 4 | blauer Hintergrund |
| 5 | grüner Hintergrund |
| 6 | roter Hintergrund |
| 7 | blinkender Hintergrund |

Ist Bit 7 gesetzt, so wird das entsprechende Zeichen nur dann blinkend dargestellt, wenn auch im Modusauswahlregister der CGA-Karte (Portadresse 3D8hex) das Blink-Bit 5 gesetzt ist. Andernfalls bewirkt das gesetzte Bit 7 einen leuchtenden Hintergrund.
Selbstverständlich lassen sich die Bits des Attributbyte beliebig kombinieren, so daß Sie nicht auf die drei Grundfarben beschränkt sind.
Das Attributbit »leuchtendes Zeichen« schaltet das Intensitätssignal am RGB-Ausgang der CGA-Farbgrafikkarte ein, damit ein Zeichen heller dargestellt wird. Ob das Zeichen auf dem Monitor aber auch wirklich heller erscheint, hängt vom verwendeten Farbmonitor ab. Nicht alle Farbmonitore nehmen nämlich das Intensitätssignal zur Kenntnis. Farbmonitore, die das Intensitätssignal auswerten, werden oftmals auch als RGBI-Monitore bezeichnet.
Tabelle 6 listet die Portadressen der CGA-Farbgrafikkarte auf. Wenn Sie diese Tabelle mit Tabelle 2 vergleichen, werden Sie feststellen, daß die Register des Monochromadapters und der CGA-Karte teilweise identisch sind. Dadurch ist für die Textmodi der beiden Bildschirmadapter ein hohes Maß an Kompatibilität gewährleistet.
Weil in den Textmodi der CGA-Farbgrafikkarte mit 80x25 beziehungsweise 40x25 Zeichen auf dem Bildschirm nur 4000 oder sogar lediglich 2000 Byte im Bildschirmspeicher der Grafikkarte benötigt werden, liegt ein Großteil der Speicherkapazität von 16 KByte brach. Dieser freie Speicherbereich wird dazu benutzt, um vier oder acht Textseiten im Bildschirmspeicher der CGA-Karte zu speichern und blitzschnell zwischen ihnen umzuschalten.
GW-Basic unterstützt dieses Verfahren beispielsweise durch den Screen-Befehl. Der dritte Parameter dieses Befehls bestimmt, in welche Textseite die nachfolgenden Basic-Ausgabebefehle schreiben, und der vierte Parameter gibt die Textseite an, die auf dem Monitor dargestellt werden soll. Während der zweite Parameter des Screen-Befehls relativ uninteressant ist, wählt der erste Parameter den aktuellen Videomodus aus. Tabelle 7 zeigt die verfügbaren Parameter für Monochromadapter, CGA-Farbgrafikkarte und EGA-Karte. Die Hercules-Grafikkarte, wird von GW-Basic, wie bereits erwähnt, nicht unterstützt.
Das BIOS im PC bietet für die Wahl der aktiven Textseite die Funktion 5 des Video-Interrupt 10hex an. Der Wert, der beim Funktionsaufruf im AL-Register übergeben wird, bestimmt, welche Textseite aktiviert wird. Die Funktion 5 macht sich dabei die Register 12 und 13 des Video-Controllers 6845 zunutze. In diesen Registern steht nämlich die Startadresse des Bildschirmspeichers, die sich mit einem Out-Befehl ganz einfach verändern läßt.
In den Grafikmodi ist der Bildschirmspeicher der CGA-Farbgrafikkarte etwas ungewöhnlich organisiert. Der Speicher ist nämlich halbiert: die erste Hälfte enthält die Bildschirminformationen für alle geraden Bildschirmzeilen (0, 2, 4 ...), die zweite Hälfte die Informationen für alle ungeraden Zeilen (1, 3, 5 ...).
Wenn Sie also den Bildschirmspeicher der CGA-Karte zur Hälfte mit dem Wert 255 beschreiben, ist jede zweite Bildschirmzeile hell. Der erste Bildpunkt der zweiten Bildschirmzeile wird demnach durch ein Byte in der Mitte des Bildschirmspeichers mit der Adresse BA000hex repräsentiert. Weil aber fast alle Programmiersprachen die Grafikmodi der CGA-Karte unterstützen, ist der direkte Zugriff auf den Bildschirmspeicher meist nicht erforderlich, so daß Ihnen der nähere Umgang mit dieser umständliche Speicherorganisation erspart bleibt.
Auch treten beim direkten Zugriff auf den Bildschirmspeicher einer CGA-Farbgrafikkarte oft Bildstörungen in Form von »Schneetreiben« auf. Im Gegensatz zu anderen Bildschirmadaptern für PCs ist der Bildschirmspeicher der CGA-Karte nämlich nicht »dual ported«, das heißt, es ist nicht ausgeschlossen, daß es bei Speicherzugriffen zu Kollisionen zwischen Video-Controller und Mi-

ORGATECHNIK KÖLN '88
INTERNATIONALE BÜROMESSE
20. bis 25. Oktober
DMV Daten- und Medien-Verlagsges. mbH
Postfach 250, 3440 Eschwege
Wir stellen aus
Halle 2.1, Gang J, Stand 60

kroprozessor kommt. Diese Zugriffs-Kollisionen machen sich als Bildstörung bemerkbar.
Das Schneetreiben auf dem Bildschirm läßt sich nur mit einer Assemblerroutine verhindern, die vor jedem Zugriff auf den Bildschirmspeicher wartet, bis der Strahlrücklauf aktiv ist. Dazu muß die Routine solange den Zustand des Bit 0 im Grafikstatusregister der CGA-Karte (Portadresse 3DAhex) abfragen, bis das Bit gesetzt ist und damit einen aktiven Strahlrücklauf anzeigt. Jetzt kann der Prozessor kurzzeitig auf den Bildschirmspeicher der Grafikkarte zugreifen, ohne Zugriffe durch den Video-Controller befürchten zu müssen.

## Die EGA-Karte stellt die Leistungen des Hercules- und CGA-Standards in den Schatten

Die EGA-Karte ist im Vergleich zur Hercules- und CGA-Grafikkarte ein wesentlich leistungsfähigerer Bildschirmadapter, denn Sie bietet eine hohe Grafikauflösung in mehreren Farben zu einem vernünftigen Preis.
So beträgt die Grafikauflösung der EGA-Karte maximal 640x350 Bildpunkte, wobei sich jeder Bildpunkt in einer von 16 Farben aus einer Palette von 64 Farbtönen darstellen läßt.
Auch emuliert die EGA-Karte alle CGA-Modi, so daß der Großteil der CGA-Software auf einer EGA-Karte läuft. Probleme gibt es allerdings bei Programmen, die die Register des Video-Controllers 6845 direkt programmieren, denn während einige EGA-Karten die Register des Video-Controllers 6845 emulieren, ist dies bei den meisten EGA-Karten aufgrund des unterschiedlichen Hardwareaufbaus nicht der Fall.
Zusätzlich ist die EGA-Karte in der Lage, die üblichen CGA-Auflösungen von 320x200 und 640x200 Punkten in 16 Fraben anzuzeigen. Diese Modi lassen sich – im Gegensatz zu den echten EGA-Modi – auch auf einem normalen Farbmonitor darstellen.
Für die Auflösung von 640x350 Punkten ist dagegen ein spezieller EGA-Monitor erforderlich, der neben der CGA-Horizontalfrequenz von 15,75 kHz auch eine zweite Horizontalfrequenz von 21,85 kHz erkennt und sich automatisch darauf einstellt.
Weiterhin läßt sich eine EGA-Karte auch an einem Monochrom-Monitor betreiben, der für den Anschluß eines Monochromadapters oder einer Hercules-Grafikkarte konstruiert ist. Eine Standard-EGA-Karte kann auf dem Monochrom-Monitor allerdings nur den Textmodus des Monochromadapters und einen zusätzlichen EGA-Monochrom-Modus mit einer Auflösung von 640x350 Bildpunkten darstellen. Manche EGA-Karten unterstützen dagegen auch den Grafikmodus der Herculeskarte mit 640x348 Bildpunkten in der Half- oder Full-Konfiguration.
Die Programmierung der EGA-Karte ist inzwischen auch kein Problem mehr, weil einerseits die Version 3.21 von GW-Basic die EGA-Karte vollständig unterstützt, und andererseits jede EGA-Karte über eine BIOS-Erweiterung verfügt, daß den Video-Interrupt 10hex um zusätzliche EGA-Funktionen erweitert.
Die direkte Programmierung des Bildschirmspeicher oder der Register lohnt sich bei der EGA-Karte allerdings in der Regel nicht, weil die Ansteuerung ziemlich kompliziert ist und die Routinen des EGA-BIOS hinsichtlich der Ausführungsgeschwindigkeit so weit optimiert sind, daß die Funktionen ausreichend schnell arbeiten. (Martin Hepp/ma)


## Impressum

**Herausgeber**
Christian Widuch
**Chefredaktion**
Andreas Hagedorn (hg), Volker Everts (ev)
**Stellvertretender Chefredakteur**
Martin Aschoff (ma)
**Redaktion**
Dieter Bäumler, (db), Gabi Börger (gb), Ulli Eike (ue), Johann Glas (gh), Eva-Maria Hierlmeier (hi), Dieter Müller (dm), Thomas Röder (tr), Andrea Scheding (as), Stefan Willmeroth (sw), Johann Zieger (zi)
**Schlußredaktion**
Eva-Maria Hierlmeier, Andrea Scheding, Andrea Voß
**Redaktionsassistenz**
Dorothea Lehner
**Koordination**
Stefan Ritter
**Produktionsleitung**
Gerd Köberich, Helmut Skoupy
**Satz**
Claudia Küllmer, Silvia Führer, Martina Siebert, Gabriela Joseph, Marcus Geppert
**Lektorat**
Susanne Mias, Christoph Dutt
**Gestaltung**
Yvonne Hendricks, Manuela Eska, Mohamed Hawa, Petra Biehl
**Reprografie**
Margarete Schenk, Dieter Schnobl
**Fotografie**
Christian Heckmann, Klaus Jatho
**Anzeigenleitung**
Wolfgang Schnell
Es gilt die Anzeigenpreisliste Nr. 2 vom 01.06. 1988
**Anzeigenverkauf**
Wolfgang Brill
**Anschrift Redaktion:**
Redaktionsbüro Everts & Hagedorn
Hans-Stießberger-Straße 3, 8013 Haar/München
Tel. (089) 46 10 01-0, Telefax (089) 46 10 01 90

**Anschrift Verlag:**
DMV – Daten & Medien Verlagsges. mbH
Postfach 250, Fuldaer Straße 6
3440 Eschwege
Telefon: (0 56 51) 80 09-0
Telex 993 210 dmv d
Telefax 0 56 51 / 80 09 33
**Verlagsbüro München**
Hans-Stießberger-Str. 3
8013 Haar/München
Leiterin Britta Fiebig
**Vertrieb**
Verlagsunion
Friedrich-Bergius-Straße 20
6200 Wiesbaden
**Druck**
Druckerei Jungfer, 3420 Herzberg
**Bezugspreise**
DOS EXTRA erscheint viermal jährlich,
Einzelpreis 18,– DM, 145 Ös, 18,– sfr.
**Bankverbindungen:**
Postscheck Frankfurt/M: Kto.-Nr.: 23043-608
Raiffeisenbank Eschwege:
BLZ: 522 603 85, Kto.-Nr.: 245 7008
**Manuskripte, Programme:**
bitte an das Redaktionsbüro Everts & Hagedorn, Redaktion DOS International, Hans-Stießberger-Str. 3, 8013 Haar/München senden.


Für unverlangt eingesandte Manuskripte und Datenträger sowie Fotos übernimmt der Verlag und die Redaktion keine Haftung. Die Zustimmung zum Abdruck wird vorausgesetzt.
Das Urheberrecht für veröffentlichte Manuskripte liegt ausschließlich beim Verlag. Nachdruck sowie Vervielfältigung oder sonstige Verwertung von Texten, nur mit schriftlicher Genehmigung des Verlages. Für veröffentlichte Programme und Basteleien übernehmen wir weder Gewähr noch Haftung. Schaltungen und verwendete Bezeichnungen müssen nicht frei von gewerblichen Schutzrechten sein. Namentlich gekennzeichnete Fremdbeiträge geben nicht in jedem Fall die Meinung der Redaktion wieder.


## Die Inserenten


Abacomp ..... 65
Albatros-Computer GmbH ..... 157
Amstrad ..... 172
Batavia ..... 28,29
Call-Soft ..... 157
Copro ..... 39
CWTG ..... 75
Data Becker ..... 58,59,68,69
DMV ..... 49,91,103,107,137,153
els ..... 123
Fecho-Soft ..... 35
Hard + Soft Computer ..... 75
Imperial Software Systems ..... 37
Individuelle Datentechnik ..... 159
Interlog ..... 15
Interest-Verlag ..... 83
Kessler ..... 65
Kirschbaum Software ..... 129
Kopisch ..... 159
K + P ..... 75
Kulenkampff + Konitzky ..... 13
MCS Kortmann ..... 139
Mental Connection ..... 171
Microsoft ..... 22,23
PC-Products ..... 43
R + P ..... 161
Softmaker ..... 2
Software Connection ..... 123
Stippler ..... 139
Vehrenberg ..... 17
Weeske ..... 163
Westcomp ..... 123
Wilski ..... 157
Ing.-Büro Zoschke ..... 139

# UniText PC

## Das »fabelhafte« Textsystem

**UniText.** Viele der Leistungsmerkmale von UniText gehören bei anderen Textverarbeitungsprogrammen in das Reich der Legenden und Sagen.
**UniText.** Intelligente Textverarbeitung. Moderne Datenfernübertragung. Neuartige Adreßverwaltung. Komfortabler Dateimanager.
**UniText.** Ein ausgereiftes Komplettpaket:

**Textverarbeitung**
- Textgröße nur durch Kapazität der Diskette/Festplatte begrenzt
- erweiterbare, automatische deutsche Rechtschreibkorrektur (30000 Worte)
- Serienbrieffunktion
- deutsche Silbentrennung
- im- und exportieren von ASCII- und WordStar-Dateien
- auch Sonderzeichen können benutzt werden
- Druckertreiber für alle gängigen Matrix-, Typenrad- und Laserdrucker

**Adreßverwaltung**
- Dateigröße nur durch Kapazität der Diskette/Festplatte begrenzt
- automatische Anredeerkennung
- Etikettendruck, Druck auf Einzelblatt- und Endlospapier
- suchen und sortieren nach allen Kriterien
- im- und exportieren von ASCII-Dateien

**Datenfernübertragung**
- 11 verschiedene Baudraten
- unterschiedliche Bildschirmemulationen (VT, ANSI, ASCII)
- senden und empfangen von Dateien
- umfangreiche Parametereinstellungen (Protokoll, Datenbits, Stopbits, Parität)
- automatische ASCII-Anpassung
- großer Empfangs- und Sendepuffer

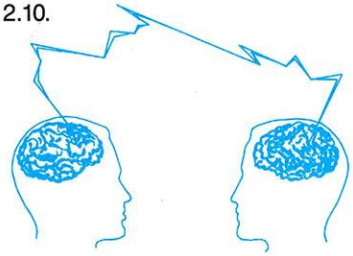
**Dateimanager**
- löschen, kopieren und umbenennen von Dateien
- ansehen von Dateiinhalten
- grafische Darstellung der Verzeichnisstruktur

**UniText** paßt sich dem Benutzer an. Gesteuert wird es kinderleicht über Pulldown-Menüs mit der Maus oder den Cursortasten oder über Control-Tasten.
**UniText.** In allen Programmteilen sind Schreibtischhilfen wie Taschenrechner, Notizblock, Uhr und Hilfsbildschirme aufrufbar.
**UniText.** Bücher schreiben und Mailboxen anrufen, Serienbriefe erstellen und Datenbanken benutzen, Festplatte organisieren und Kundendateien erstellen, Diktate korrigieren und Bewerbungen verschicken, Diplomarbeit schreiben und Vereinsmitglieder verwalten, Zeitungen machen und, und, und ...
**UniText.** Das "fabel"hafte Textsystem. Mit ausführlicher deutscher Bedienungsanleitung.
**UniText.** Für Hausfrauen, Manager, Journalisten, Studenten, Sekretärinnen, Beamte, Rechtsanwälte, Ärzte, Lehrer, Übersetzer, Ingenieure, Geistliche, Spediteure, Selbständige, Einzelhändler ...
**UniText.** Lauffähig auf allen gebräuchlichen Grafikkarten. Für jeden IBM-kompatiblen PC, XT oder AT mit mindestens 256 KByte Hauptspeicher, mindestens einem Diskettenlaufwerk und MS-DOS oder PC-DOS ab Version 2.10.

Händleranfragen erwünscht

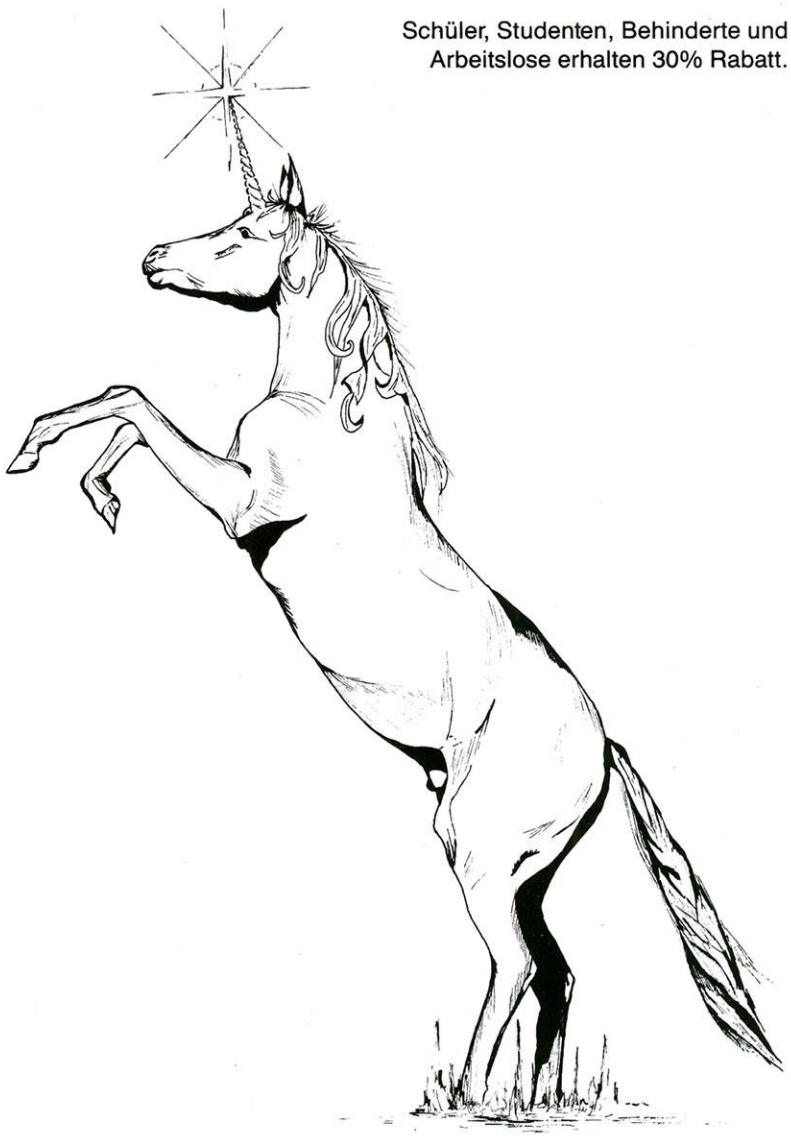

**mental connection**

Althaus und Zietlow GbR
Schwelmer Straße 119
D-5820 Gevelsberg
Telefon: 0 23 32/5 14 43

Das Unternehmen für innovative Computertechnik

Schüler, Studenten, Behinderte und Arbeitslose erhalten 30% Rabatt.



***Für märchenhafte***

# 99,– DM

Bitte Coupon ausschneiden und an nebenstehende Adresse schicken

DOS E 4/88

○ Bitte senden Sie mir ____ Exemplare von UniText PC
○ per Nachnahme ○ per Vorausscheck
○ Ich benötige nähere Informationen zu UniText PC

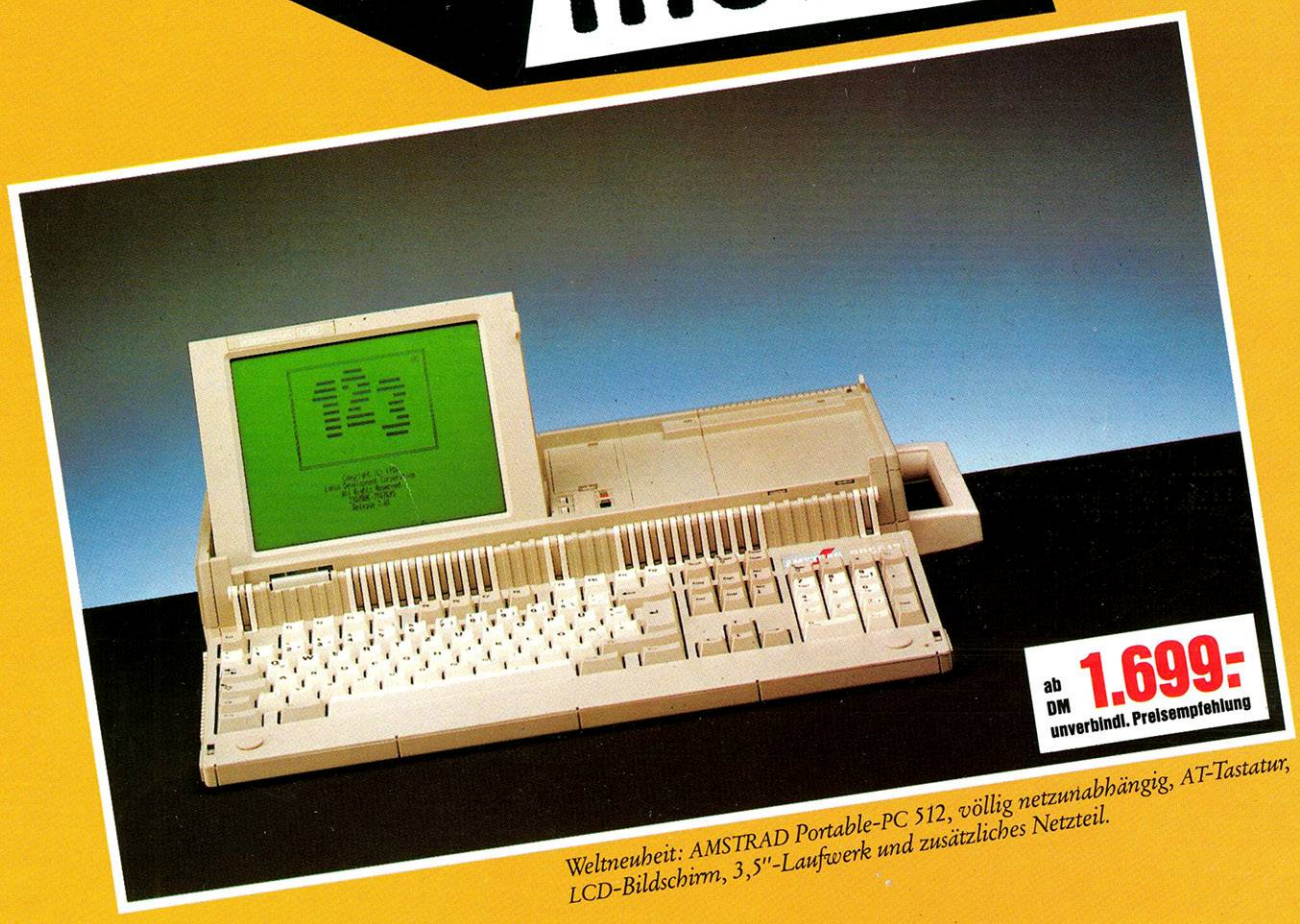
Name

Straße

Ort

Datum, Unterschrift

# Intelligenz macht mobil.




*Weltneuheit: AMSTRAD Portable-PC 512, völlig netzunabhängig, AT-Tastatur, LCD-Bildschirm, 3,5"-Laufwerk und zusätzliches Netzteil.*

Jeder „Standard" ist nur das, was man aus ihm macht.
Zum Beispiel AMSTRAD: Da gibt es jetzt den ersten PC am Griff – ohne lästiges Netzkabel. Damit ist ein vollwertiger PC so unabhängig wie ein Batterie-Rasierer. Nur einer von vielen Beweisen für die Ideen und die Leistung von AMSTRAD-Computern.

Über 1 Million Computer von AMSTRAD arbeiten bereits auf Europas Schreibtischen. Weil sie einfach zu bedienen sind. Weil Qualität und Preis stimmen. Und weil Service und Beratung von AMSTRAD keinen alleine lassen.

## So hätten Computer von Anfang an sein sollen.

- PC 1640 ab 1.699,– DM. PC 1512 ab 1.299,– DM.
- Portable-PC 512 ab 1.699,– DM.
- Textsysteme: PCW 9512 für 1.699,– DM. PCW 8512 ab 999,– DM.
- Semi-professioneller CPC 6128 ab 799,– DM.
- 9-Nadel-Drucker ab 599,– DM. 24-Nadel-Drucker ab 899,– DM.

(unverbindliche Preisempfehlungen)
Jetzt beim namhaften Fachhandel.

AMSTRAD

**Computer sind für jeden da.**

Händler- und Produkt-Informationen bei AMSTRAD GmbH, Abt. VKF
Robert-Koch-Straße 5, 6078 Neu-Isenburg